

**AN AGENT-BASED VARIOGRAM MODELLER: INVESTIGATING
INTELLIGENT, DISTRIBUTED-COMPONENT GEOGRAPHICAL
INFORMATION SYSTEMS**

ABDULLAH AL-ZAKWANI

Ph.D.

2013

DECLARATION

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

ABSTRACT

Geo-Information Science (GIScience) is the field of study that addresses substantive questions concerning the handling, analysis and visualisation of spatial data. Geo-Information Systems (GIS), including software, data acquisition and organisational arrangements, are the key technologies underpinning GIScience. A GIS is normally tailored to the service it is supposed to perform. However, there is often the need to do a function that might not be supported by the GIS tool being used. The normal solution in these circumstances is to go out and look for another tool that can do the service, and often an expert to use that tool. This is expensive, time consuming and certainly stressful to the geographical data analyses. On the other hand, GIS is often used in conjunction with other technologies to form a geocomputational environment. One of the complex tools in geocomputation is geostatistics. One of its functions is to provide the means to determine the extent of spatial dependencies within geographical data and processes. Spatial datasets are often large and complex. Currently Agent system are being integrated into GIS to offer flexibility and allow better data analysis. The theis will look into the current application of Agents in within the GIS community, determine if they are used to representing data, process or act a service.

The thesis looks into proving the applicability of an agent-oriented paradigm as a service based GIS, having the possibility of providing greater interoperability and reducing resource requirements (human and tools). In particular, analysis was undertaken to determine the need to introduce enhanced features to agents, in order to maximise their effectiveness in GIS. This was achieved by addressing the software agent complexity in design and implementation for the GIS environment and by suggesting possible solutions to encountered problems. The software agent characteristics and features (which include the dynamic binding of plans to software agents in order to tackle the levels of complexity and range of contexts) were examined, as well as discussing current GIScience and the applications of agent technology to GIS, agents as entities, objects and processes. These concepts and their functionalities to GIS are then analysed and discussed. The extent of agent functionality, analysis of the gaps and the use these technologies to express a distributed service providing an agent-based GIS framework is then presented.

Thus, a general agent-based framework for GIS and a novel agent-based architecture for a specific part of GIS, the variogram, to examine the applicability of the agent-oriented paradigm to GIS, was devised. An examination of the current mechanisms for constructing variograms, underlying processes and functions was undertaken, then these processes were embedded into a novel agent architecture for GIS. Once the successful software agent implementation had been achieved, the corresponding tool

was tested and validated - internally for code errors and externally to determine its functional requirements and whether it enhances the GIS process of dealing with data. Thereafter, its compared with other known service based GIS agents and its advantages and disadvantages analysed.

<u>Table of Contents</u>	<u>Page</u>
Abstract	v
Acknowledgements	xv
Dedication	xvi
List of Abbreviations	xvii
 Chapter One: Focus of Investigation.	 1
1.1 Introduction	1
1.2 Problem Domain and Research Focus	5
1.3 Aims and Objectives	13
1.4 Structure of the Thesis	14
 Chapter Two: Intelligent, Distributed Component GIS	 17
2.1 Introduction	17
2.2 GISystem, Science and Engineering	18
2.3 Some Key Research Themes	22
2.3.1 GIS and Environmental Modelling	22
2.3.2 Spatial Data Mining	24
2.3.3 Geocomputation and Geosimulation	27
2.3.4 Web 2.0	29
2.3.5 Ubiquitous GIS	32
2.4 Interoperability and Tool Coupling	34
2.4.1 External Coupling of GIS	33
2.4.2 Interoperability	34
2.4.3 Tool Coupling	37
2.4.4 Coupling GIS for Better Data Analysis	39
2.5 Intelligent, Distributed Component GIS	40
2.5.1 Rationale of Distributed GIS Components	41
2.5.2 Developing Intelligent Capability for Distributed GIS Components	43
2.5.3 Research Structure of Agent-based Distributed Component GIS	48

Chapter Three: Agent-based Technologies	51
3.1 Introduction	51
3.2 Characteristics of Software Agents	51
3.3 Some Applications of Software Agents	58
3.4 Use of Agent is Geosimulation	66
3.4.1 Early Use of Cellular Automata Geosimulation	66
3.4.2 Next Stage of Using Agents in Geosimulation	69
3.4.3 Issues in the Use of Agents for Geosimulation	72
3.5 Classification of Agent-based Applications in the Spatial Domain	74
3.5.1 Understanding Agency	74
3.5.2 Classifying Agent Usage in the Spatial Domain	77
3.6 Demands and Gaps in Deploying Agent Technology in GIS	83
 Chapter Four: Modelling the Variogram	 87
4.1 Introduction	87
4.2 Geostatistics	85
4.3 The Variogram	93
4.3.1 Modelling the Variogram	96
4.3.2.1 Unbounded	96
4.3.2.2 Bounded	98
4.4 Data Aspects of Modelling the Variogram	98
4.4.1 Data Sampling	99
4.4.2 Polishing and Outlier Removal	100
4.4.3 Declustering	109
4.5 Cross Validation and Goodness of Fit	110
4.6 Kriging	112
4.7 Conclusion	113
 Chapter Five: Methods for Implementing and Testing an Agent-based Variogram Modeller	 114

5.1	Introduction	114
5.2	Agent Development Methodology	114
5.2.1	Tropos	114
5.2.2	From Design to Implementation	118
5.3	The Programming Environment	119
5.3.1	A Proposed Agent Programming Language Structure	125
5.3.2	Agent Communication Infrastructure	133
5.3.3	Critique of Current Agent Programming Language	134
5.3.4	Guidelines for Producing a Learning-friendly Programming Language	143
5.4	Data Sets for Testing an Agent Variogram Modeller	145
	Chapter Six: The Structure of the VAC	149
6.1	Introduction	149
6.2	GIS Agent-based System Architecture	149
6.2.1	The Distributed Agent Framework for GIS Agents	150
6.2.2	The Platform for Intelligent Distributed Component GIS	152
6.3	VAC Design Using Tropos	154
6.3.1	Data Finder	162
6.3.2	Integrity Checker	163
6.3.3	Data Analyser	165
6.3.4	Mathematical Modeller	167
6.3.5	Sample Modeller	168
6.3.6	Strategy Comparer	170
6.3.6	Model Fitter	171
6.3.7	Model Plotter	173
6.4	Algorithm, Process and Data Handling for the VAC	173
6.4.1	Data Handling of the VAC - Data Responsibility Diagram	176
6.4.2	Agent Architecture: Design of VAC using Unified Agent Diagram	177
6.5	Required Features for Software Agents as Components for GIS	178
6.5.1	The Agent Reproduction Capability	182

6.5.2 The Agent Reproduction	183
6.5.3 The Dynamic Binding Mechanism	187
6.6 The Agent Communication	191
6.7 The Agent Learnability	195
6.8 Conclusion	198
 Chapter Seven: Results and Analysis	 200
7.1 Introduction	200
7.2 The Agent Functionality Test	200
7.3 Internal Validation	204
7.3.1 Non-response Problem	204
7.3.2 Large Dataset Problems	207
7.4 External Validation	209
7.4.1 Scenarios: Testing the new agent features on VAC System	210
7.4.2 Scenario 1: Testing Dynamic Binding and Reproduction functions	211
7.3.2.1 Experiment Run 1	212
7.4.3 Scenario 2: Testing Communication and Faulty Data	217
7.3.2.1 Experiment Run 2	218
7.5 The Agent Handling the Walker Lake Data	223
7.5.1 Variogram Selection Process	227
7.5.1 The Variogram Process	229
7.6 Comparing the VAC to Other 'Service' Based Agent Technologies in GIS	233
7.7 Conclusion	238

Chapter Eight: Conclusions and Future Research	239
8.1 Introduction	239
8.2 Operational Functionality of the VAC	240
8.3 Intelligence Focus in the VAC	241
8.4 Suitability of Agent Characteristics to GIS	243
8.5 Current GIS Issues and Improvements Through Agents: Geostatistical Pilot Study	245
8.6 Critical Evaluation and Conclusion	248
8.6.1 Significance and Implication of the Findings	248
8.6.2 Critical Analysis	250
8.6.3 Conclusion	252
8.7 Future Work	255
8.7.1 Use of the VAC With Other Agent Systems in GIS	257
8.7.2 Communication of Distributed Component in GIS	259
8.7.3 Expanding Distributed Component in GIS	269
References and Bibliography	271
Appendix A	319
Appendix B	326
Appendix C	338
Appendix D	349
Appendix E	354
Appendix F	357
Appendix G	359

<u>List of Tables</u>	<u>Page</u>
Table 2.1 Search Patterns for Classic Data Mining and Spatial Data Mining	26
Table 3.1 Examples of Utilising Agents in the Spatial Domain	81
Table 5.1 Explanation of the Walker Lake Data	146
Table 6.1 Agency Application on VAC	181
Table 7.1 Comparison of Agents Run	235
Table 8.1 Knowledge Structure or Agent System	256

	<u>List of Figures</u>	<u>Page</u>
Figure 1.1	Example of a Variogram	11
Figure 2.1	The Ubiquitous Computing Environment for GIS	33
Figure 2.2	Research Structure for Distributed Agent Platform	51
Figure 3.1	Demonstration of Cellular Automata	68
Figure 3.2	Proposed Classification of Utilising Agents in the Spatial Domain	80
Figure 4.1	Variogram, Unbounded Model	98
Figure 4.2	Variogram, Bounded Model	100
Figure 4.3	Box Plot	104
Figure 5.1	Comparison of Agent Methodologies and Functionality	117
Figure 5.2	BDI Agent Functionality and Message Passing on a Class Diagram	124
Figure 5.3	Agent Diagram - Structure for a Top View of Agent Representation	126
Figure 5.4	Connection Between Two Agent Diagrams, Showing Benefit	128
Figure 5.5	Depiction of Action for an Agent	129
Figure 5.6	Depiction of Reaction by Agent	130
Figure 5.7	Structure of Agent Implementation	131
Figure 5.8	Mapping Structure of Agent: Tropos Methodology to Class Object-oriented	132
Figure 5.9	UML Class Diagram	136
Figure 5.10	Translation of a UML Class Diagram into Object-oriented Program Code	136
Figure 5.11	Code Structure for Agent Diagram	137
Figure 5.12	Actor that has Goals and Beliefs	139
Figure 5.13a	Coding Structure of Agent on the JACK Platform	140
Figure 5.13b	Increased Complexity of Coding Structure on the JACK Platform	140

Figure 5.13c	Coding a Sub-class Which Increases Complexity for Coding an Agent on the JACK Platform	140
Figure 5.14a	Representation of the Structure for the Proposed Programming Environment	141
Figure 5.14b	Mapping Code Generated from Figure 5.15a	142
Figure 5.16	JACK Programming Environment	144
Figure 5.17	Concentration and Clustering of V Values in the Walker Lake Data	148
Figure 6.1	Architecture Design for Component base GIS, Showing Three Tier Structure	150
Figure 6.2	Fundamental Protocol which All Agent Components Must Adhere to Within Distributed Component GIS	152
Figure 6.3	General Architectural Platform for GIS Agent Infrastructure	153
Figure 6.4	Actor Diagram for the GIS System	155
Figure 6.5	Goal Diagram of the Geostatistition Actor	156
Figure 6.6	Goal Diagram of the Variogram Actor	157
Figure 6.7	Actor Decomposition of the Variogram Agents	158
Figure 6.8	Variogram Interagent Dependence-Dependent-Dependee Diagram: Internal Variogram Actors and their Dependencies	160
Figure 6.9	Algorithm for Variogram Agent	161
Figure 6.10	Goal Analysis for the Data Finder Agent	162
Figure 6.11	Integrity Checker Agent Diagram Using Tropos Methodology	164
Figure 6.12	Data Analyser Agent Diagram Using Tropos Methodology	166
Figure 6.13	Mathematical Modeller Agent Diagram Using Tropos Methodology	167
Figure 6.14	Sampler Agent Diagram Using Tropos Methodology	169
Figure 6.15	Sampling Action Flow Diagram	170
Figure 6.16	Model Fitter Agent Diagram Using Tropos Methodology	172
Figure 6.17	Agent's Plan and Event Flow	175
Figure 6.18	Data Flow Within Agents and Their Plans	176
Figure 6.19	VAC Design Using proposed Unified Agent Diagram	178

Figure 6.20	Activity Diagram Representing Algorithm for Reproduction	184
Figure 6.21	Agent Reproduction Mechanism	186
Figure 6.22	Structure of Dynamic Binding of Plans	188
Figure 6.23	Dynamic Binding of Plans Console Design	189
Figure 6.24	Activity Diagram Representing Algorithm for Dynamic Binding	190
Figure 6.25	Activity Diagram Representing Algorithm for Communication	192
Figure 6.26a	Communication Flow Without Utilising VAC Functionality	193
Figure 6.26b	Communication Flow Utilising VAC Functionality	194
Figure 6.20	Activity Diagram Representing Algorithm for Learnability	196
Figure 7.1	Process that Takes Place During Agent Analysis	201
Figure 7.2a	Logs Showing Messages for Misplaced File	205
Figure 7.2b	Logs Showing File That is None Responsive	206
Figure 7.3	Analysis Being Rejected by ArcGIS due to the Dataset Size	208
Figure 7.4	Reasoning Algorithm	211
Figure 7.5	Report of File Reception by the Agent System	212
Figure 7.6	Cluster Identified by Agent	213
Figure 7.7	Printed File of Cluster Sample	213
Figure 7.8	Printed File Representing the Outlier Found by the Agent	214
Figure 7.9	Dynamic Binding Dialog Box	215
Figure 7.10	Quadratic Trend to be Handled	215
Figure 7.11	Agent Announce New Agent Reproduced	216
Figure 7.12	Experimental Variogram Fitted and the Derived Theoretical Model as Spherical	217
Figure 7.13	Announcement to human User	219
Figure 7.14	The trace Function of VAC, Communication	219
Figure 7.15	Message Show an Agent Announcing o Corrupt File	220
Figure 7.16	Faulty File Announced and Both Human and Agent Users Have the Possibility to Re-Examine	220
Figure 7.17	Agent Receive Large File and Splits it The Place New File to the Examination Folder	221

Figure 7.18	Experimental Variogram Fitted and Agent Derive	
	Theoretical Model as Gaussian	221
Figure 7.19	Cluster Identified by Agent	222
Figure 7.20	Linier Trend to Be Handled	222
Figure 7.21	Experimental Variogram Fitted and Agent Derive	
	Theoretical Model as Gaussian	223
Figure 7.22	Identified Clusters	224
Figure 7.23	Clusters Identified in Walker Lake Data	226
Figure 7.24a	Anisotropic Experimental Variogram at 45 degree angle	227
Figure 7.24b	Anisotropic Experimental Variogram at 90 degree angle	227
Figure 7.25a	A Spherical isotropic Variogram Determine by the Agent	
	Before Remove Trend	228
Figure 7.25b	A Gaussian isotropic Variogram Determine by the Agent	
	After Remove Trend	228
Figure 7.26a	Kriging Map of Walker Lake Data as Identified by the VAC	229
Figure 7.26b	Kriging Map of Walker Lake Data (Isaaks and Srivastava, 1989)	230
Figure 7.27a	Map of Walker Lake Data Showing Residual Errors from	
	Kriging Data by VAC	231
Figure 7.27b	Map of Walker Lake Data Showing Residual Errors from	
	Kriging Data by Isaaks and Srivastava, 1989	231
Figure 7.29	Exhibition of the Data Analyser Agent	235
Figure 7.30	Result from Oracle and Repast	236
Figure 7.31	Comparing Agent Performance	237
Figure 8.1	Agent as Service Provider to Geosimulations	258
Figure 8.2	Agent Communication and Information Sharing	260

ACKNOWLEDGEMENTS

I would like to thank the many friends, relatives and supporters who have made this happen. I would like to thank my loving wife, Kristel, with her emotional support and for putting up with all the late night noises throughout my research. Professor Allan J. Brimicombe for his help and encouragement, and to Harlambous Mouratidis, Yang Li and all my friends and family who made this happen with their invaluable help.

Finally, to my loving parents for their love, support and guidance.

DEDICATION

This is dedicated to my loving wife Kristel
and my amazing parents and siblings; God bless them all.

LIST OF ABBREVIATIONS

ACL	Agent Communication Language
AI	Artificial Intelligence
ALGAs	Artificial Life Geospatial Agents
AMS	American meteorological Society
AOSE	Agent-Oriented Software Engineering
APA	American Psychologist Association
API	Application Programming Interface
APL	Agent Programming Language(s)
AUML	Agent Unified Modelling Language
B.L.U.E	Best Linear Unbiased Estimator
BA	Basic Automata
BAC	Broker Agent Component
BDI	Belief, Desire, Intention (agent architecture)
CA	Cellular Automata
CMs	Contingency Management System
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DAA	Data Analyser Agent
DAI	Distributed Artificial Intelligence
DB	Database
DBA	Data Broker Agent
DBMS	Database Management System
DCOP	Distributed Constraint Optimization Problems
DEMs	Digital Elevation Models
DESIRE	DEsign and Specification of Interacting REasoning components
DPS	Distributed Problem Solving
DS	Distributed Systems
ESDA	Exploratory Spatial Data Analysis
FPS	First Person Shooters
GI-Engineering	Information Science Engineering
GIS	Geographical Information System(s) (being the combination of both GIScience and GISystems)
GIScience	Geographical Information Science
GML	Geographic Mark-up Language
GPS	Geographical Positioning System
GUI	Graphical User Interface
GWR	Geographically Weighted Regression

HCI	Human-Computer Interaction
ICA	Integrity Checker Agent
ICTs	Information Communication Technologies
IDC	Intelligent Distributed Component
JPL	Jet Propulsion Laboratory
JVM	Java Virtual Machine
KDD	Knowledge Discovery in Databases
LBS	Location Based Service
MAC	Map Agent Component
MAS	Multi-Agent Systems
MBR	Minimum Bounding Rectangle
MDL	Model Definition Language
MFA	Model Fitter Agent
ML	Machine Learning
ML	Machine Learning
MMA	Mathematical Modeller Agent
MPA	Model Plotter Agent
NCIC	National Cartographic Information Centre
ODBC	Open Database Connectivity
OGC	Open Geospatial Consortium (previously known as OpenGIS Consortium)
OLAP	On-Line Analytical Processing
OO	Object-Oriented
OOP	Object-Oriented Programming
OpenGIS	Open Source Geographical Information Systems
PAI	Parallel AI
PL/SQL	Procedural Language/Structured Query Language
SA	Sampler Agent
SCA	Strategy Comparer Agent
SGAs	Software Geospatial Agents
SMS	Short Messaging System
SSM	Summary Schemas Model
TAC	Trading Agent Competition
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modelling Language
VAC	Variogram Agent Component(s)
WNV	West Nile virus

CHAPTER ONE: FOCUS OF INVESTIGATION

1.1 Introduction

The study of the Earth, its geographical resources and natural or man-made phenomena, is recognised as being important in human development. Environmental impacts, evident from recent catastrophes like Hurricane Katrina which devastated New Orleans and the Mississippi area in 2005 and the tsunami which battered the Indian Ocean shores in 2004, have reinforced the need to better model and understand geographical and environmental phenomena. Furthermore, the human race can be seen as the perpetrator of some environmental catastrophes, including those linked to recent climate change, and has brought to the fore issues relating to water and soil contamination, soil erosion, sedimentation and deforestation, and the social and economic impact these have on communities. Most of these issues are fundamentally spatial; being to varying degrees geographically interrelated. For example, the evident pattern of land use in relation to availability of water, energy consumption in relation to environmental changes and global warming, industrialisation to water and soil contamination, and even the shift of large industries from Europe to Asia provides evidence of cause and effect of spatial relationships. The consequences of these interrelationships and spatial dependencies in geographical information (GI) cannot be ignored. Skupin and Fabricant (2008) have defined the work of GI scientists as generally to “investigate the conceptualization, analysis, modelling and depiction of geographical phenomena and processes with regard to geographical scale”.

To identify and understand the spatial issues and relationships in modelling and understanding the environment in which we live, it is essential that pertinent data are recorded, organised and processed. Using large amounts of data to model such complex phenomena and their interrelationships would not be possible to any significant degree without information technology (IT).

Information processing and computing has become ubiquitous since the advent of PC and the Internet. According to Moore's law, computer capacity is on an exponential increase. This includes computing memory, processing and communication capacity, which has also been reinforced by the declining cost of storage and data processing. Moreover, since the introduction of the Internet, communication with computers worldwide has become cheap and easy to attain. Thus, the computing world has moved from a centralised system, where one powerful computer performed many queries, to a distributed computing system where many smaller processors are used to perform a large number of queries. This concept is known as 'distributed processing computing', where multiple systems, remote to each other, are used to perform functions as a single system (Wooldridge, 2003). Distributed processing computing is seen to be the solution to systems that handle large datasets, complex mathematical solutions and intensive graphical visualisation (Peng and Tsou, 2003; Wu *et al.*, 2004; Yang and Raskin, 2009) and has inspired considerable research and development. On the other hand, the Internet has provided an environment conducive to data and information storage and sharing. This represents a jump from the pre-Internet data-poor era where data were only available on an individual user basis, to the now post-Internet era with strong evidence of data richness (O'Reilly 2005), where most information is posted on the Web for easy access (on a free or payment basis) for everyone to use.

With the processing increment and data availability, it became easier to integrate technologies and information to better handle and understand complex geographical problems (Goodchild, 2003; Goodchild *et al.* 2004), especially with more mature technologies such as Geographic Information Systems (GIS), Geographical Positioning System (GPS), remote sensing and the formation of the Open Geospatial Consortium (OGC; www.openspatial.org). GIS are "a powerful set of tools for collecting, storing, retrieving at will, transforming and displaying spatial data from the real world for a particular set of purposes" (Burrough and McDonnell, 1998). GIS offer flexibility of scale on mapping and visualising information (Atkinson and Tate, 2000).

Thus, the main difference between hand-drawn maps (or any other map drawn with non-GIS tools) and GIS-produced products is that using GIS one can link and overlay different layers to use features from and analyse spatial relationships based on these different layers. These layers could even have been compiled from different data sources at different scales, though the overlay of such diverse layers may have quality implications (Brimicombe, 2003). However, the main strength of GIS is their ability to utilise many specialised tools for different functions, such as for data sorting, data visualisation and even the mathematical modelling of patterns within the data. This, together with data integration and their ability to handle and deal with large datasets, can provide an enabling environment towards solving many geographical problems. However, some weaknesses exist for GIS, which include their general lack of ability to integrate separate tools effectively to work on the same problem without having an expert user to physically facilitate the exchange of data from one tool to another. This requires expert users who know how to effectively utilise spatial data and the available heterogeneous tool sets. In this case, Goodchild (2006: p1) has stated that “too few people are sufficiently trained to use GIS tools effectively”, specifically with reference to the data, but emphasis should also be placed on the number of available tools one expert can use well. Many proprietary or even free tools are available on the Internet and industrial Intranets which are capable of providing good quality analysis, but due to a lack of knowledge and the ability of any one expert to master them all, it is not possible for this advantage to be fully utilised. Tool interoperability is an important area of the GIScience research agenda (Brimicombe, 2003; Mark, 2003; Goodchild, 2008), more so today across networks (Yang and Raskin, 2009). Interoperability has been defined as the ability of different components of a system to work together, having a unified understanding of the information structure which can then be dealt with by each component as a service to one another (Albrecht, 1996). Creating and joining together a more distributed structure to achieve this in using GIS, as opposed to the traditional monolithic GIS, continues to need urgent attention.

Distributed computing has at its heart the optimal utilization and allocation of computing resources (Yang and Ruskin, 2009), thus in computer science, software agents in distributed computing are being studied as they offer flexibility for information handling and complex system development (Bigus and Bigus, 2003; Callan, 2003).

Agents have been defined by Weiss as:

“...a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives” (1999: p29).

Franklin and Graesser define agents that exhibit intelligence as being

“...a system situated within an environment, senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future” (1996: 6).

Software agents are entities that exhibit certain characteristics (which could include intelligence and the ability to work autonomously). They have foundations in software development and artificial intelligence (AI) in which (in the latter case) an agent would have knowledge (what it knows about both itself and its environment) and may be able to make decisions as to a course of action to take. They form the fundamental concept for Distributed Artificial Intelligence (DAI) systems, which at its core employs agents connected via a networked environment, thus enabling collaboration, to achieve a common goal (Nwana 1996). This concept also underlies the principle for the mechanism of Distributed Problem Solving (DPS), where each agent does not have enough knowledge to solve the problem on its own, but through collaboration with other agents they are able to solve the problem together (Potok *et al.* 1999; Jennings and Wooldridge 2000; Lee *et al.* 2003). This also has the advantage of parallel processing gained from distributed systems, where each agent can work at the same time as opposed to one after another, thus greatly reducing the overall processing time (O'Hare and Jennings, 1996). Thus, software agents offer a considerable potential for overcoming the types of tool and data interoperability problems that are of interest to GI scientists. This will be the focus of the investigation presented in this thesis.

1.2 Problem Domain and Research Focus

A 'numerical model' refers to a process that endeavours to find an analytical solution to a problem and thereby be able to predict the behaviour of the system from a given set of parameters and initial conditions. For example, with reference to oceanography and meteorology, AMS (2006) defines numerical models as "the prediction of flow evolution via numerical construction of approximate solutions to the governing equations". In computing terms, numerical models are viewed as simulation processes which can combine with reality (actual events) responses for any set of provided inputs. These kinds of simulations have been extensively used, simulating the first Gulf War in Kuwait for example, by modelling the war zone terrain for some 66,239 tanks, trucks and other military activities (JPL, 1999). Using numerical simulations, many geographical hazards can be studied and identified to prevent, or provide warnings of, possible disasters. For example, Chua *et al.* (2002) have utilised a numerical simulation to understand a particular flooding process and consequently provide a preventive mechanism against the damage such floods cause to crops.

Recent computing usage for numerical modelling has seen a shift from tightly coupled monolithic systems to loosely coupled distributed component-based systems. This has been due to the availability of a large quantity of resources (such as lightly used computer terminals) residing remotely from each other, which can be merged together using the networking infrastructure provided by the Internet. Currently, distributed processing technologies are at the forefront of computing research and development (Bigus and Bigus, 2003; Callan, 2003; Wooldridge, 2003; Goodchild *et al.*, 2004; Rey, 2009). The distributed processing technique is also associated with distributed components computing, where multiple systems, remote to each other, are used to perform functions as a single system (Wooldridge, 2004; Xu *et al.* 2008).

In software development, a paradigm has emerged in which software agents are seen to be the main actors as distributed components, where these agents play an important role in cementing the shift from monolithic programs to distributed

architectures. The current architecture of GIS software is mostly based on a monolithic approach geared to perform a specific set of functions. This is now being improved through the aid such as Application Programming Interfaces (APIs), which are a separate set of functions that can be integrated into software without changing the original functionality of that software, thus allowing the integration of other tools into existing GIS software. In GIS, these APIs include additional analytical functions, which allow for more complex functions to be performed on the data. For example, in ArcGIS, a range of APIs can be used to build extra functions (applications) to support geostatistics (<http://resources.esri.com/arcgisservier/apis/>). Geostatistics Analyst extension of ArcGIS provides the ability to determine the spatial dependency and interrelation of values in a dataset (Johnston *et al.*, 2001). Given the size (normally very large) of a geographical dataset and the complex features it represents, these tools are designed to consume considerable CPU power so as to be able to process the given information. This resource requirement has to be established according to the resource availability on a single computing machine, and thus most of these tools have a predefined dataset size which they can handle on a single computer (for the ArcGIS Geostatistical Analyst this is up to 300 data points only). This is due to the system being monolithic and resource intensive, requiring possibly many hours to perform its analysis on one dataset.

In terms of programming, GIS software was initially developed using structured programming languages like FORTRAN to produce modular toolkits, then later moved on towards a more robust object-oriented programming (OOP) paradigm. GIS software, based on OOP, provide a basic level of interoperability. Interoperability can be viewed as:

“the cross-platform use of computer applications, i.e. the ability to use applications in different computer hardware environment and operating systems without any need to change the programs and re-train the users” (Pang-Lo and Yeung, 2006:132).

Interoperability in GIS conveys the same meaning as in software development as defined by Troelsen (2002), Nathan (2002) and Bukovics (2006).

Furthermore, Albrecht (2005) suggests developing loosely coupled GIS, where components can perform functions on data that do not necessarily reside on the same computing machine, but which could be accessed only when needed. A number of studies already exist which have tried to utilise this architecture, like those discussed by Peng and Tsou (2003) and Wu et al. (2004). However there are often limitations, such as: the high CPU cost of processing spatial data; the lack of authentication and authorisation of web service users for GIS; no clear definition of structure and methods for data compression; no obvious support for the complicated GIS workflows. Due to these issues, many problems arise in coupling numerical simulations to GIS software. Loosely coupled technology emphasizes on having components to connect services using interfaces to enhance Service-Oriented Architectures which can be robust in that change in one service does not require changes in all associated services. Loose coupling can be regarded as an:

“approach to the design of distributed applications that emphasises agility (the ability to adapt to changes)”, and furthermore “intentionally sacrifices interface optimization to achieve flexible interoperability among systems that are disparate in technology, location, performance, and availability....which enhances reusability” (Kaye, 2003: p 32).

Web services and delivery are the example of using loosely coupled technologies (Nathan, 2002; Bukovics, 2006). Web services are often seen as “implementation of capabilities for access by other application (or other web services) via industry standard network” (Chatterjee and Webber, 2004). To associate this with the loose coupling as described by Kaye (2003) and Chatterjee and Webber (2004) explain that “an application can use the capabilities of a Web service by simply invoking it across a network without having to integrate it” which represents the concept of reusability. Here the relation of loosely coupled and Web delivery is entangled when the loosely

coupled components are communicating through the network and one of the example where this phenomenon occur is on the Web services.

To emphasise the above, Brimicombe (2003) discusses the efficient interoperability and integration of GIS tools as one of the required and important aspects in GIScience research. He points out the possible issues that arise when coupling environmental simulation modelling to GIS, including data quality and the choice of algorithm to be used which are compounded by the architecture of GIS software, with issues concerning scales and residual uncertainty being other effects present in analytical decision-making. The availability of computing resources and networking technologies (like the Internet) is seen as an avenue to increase the performance of GIS technology (Albrecht 2005, 2005; Ali and Moulin, 2005; O'Reilly, 2005). The current issue is that most GIS software can be viewed as too monolithic, too heavy and thus usually reside within one single machine. Nor are they easily personalised to allow light communication over a network. Thus, there remains a need to devise a structure where GIS functions and other tools can be integrated for optimum performance. Distributed components computing in GIS is seen as one potential solution to achieve this goal.

Regarding spatial data, researchers such as Diggle (1983) have emphasised that anything residing in a space (any space surface) is a potential candidate for being a variable to be studied. Furthermore, reference to Tobler's first law of geography (Tobler, 1970), which states that "everything is related to everything else, but near things are more related to each other", expresses the basic principle of spatial dependence or spatial autocorrelation within geography and the environmental sciences. Tobler's statement also expresses one of the fundamental issues of GIScience and how it is desirable to identify the extent of dependency within a dataset, in order to identify interesting (socially or physically beneficial) features and events, with their knock-on effects.

A specialist branch of spatial analysis, known as geostatistics, is dedicated to analysing geographical data having some degree of spatial dependence or autocorrelation (de Smith *et al.*, 2007). In geostatistics, unlike conventional mathematical statistics, the data represent geographical locations, are thus interlinked and related due to their location and often do not conform to the assumption inherent in standard statistical procedures. Location is expressed as coordinates $\{x, y\}$ on the plane (\mathfrak{R}), and with further attributes of location.

Spatial data are often large datasets and, which inevitably contain errors that may propagate themselves to the final analysis of a study. To analyse the data, like the traditional statisticians who used formulae and graphs of different kinds, geostatistics also utilise similar tools to identify quality issues, and to define and estimate the relationship and dependency within a dataset. The geostatistical tool that models the spatial dependency of variables is known as a variogram, a graph of the cumulative relative variance for increasing distance (see Figure 1.1). A variogram is explained by Isaaks and Srivastava (1989) as being “half the average squared difference between the paired data values” and denoted by γ . It is used to clarify and identify the important features of the phenomena under study (Saldaña *et al.*, 1998).

In many circumstances, the phenomenon being modelled is not sampled densely enough to give a sufficiently detailed array of data, so additional points need to be predicted to provide extra values at unknown points. Geostatistics also plays an important role in this prediction. One of the geostatistical tools that help achieve this prediction is kriging. Introduced by Matheron (1963) from algorithms defined in Krige (1951), it is explained in Olea as:

“a collection of generalized linear regression techniques for minimizing an estimation variance defined from a prior model for covariance” (1991: p26).

Pringle (1980) explains that one of the objectives of geography as a discipline “is to develop explanatory theory of cause and effect” on the spatial processes or events which, by use of techniques such as the variogram and kriging, could be achieved. Thus, geostatistics from this perspective can be taken as interpolation techniques used to handle the issue of relation and prediction of attributes within a spatial plane (\mathfrak{R}) and their relation to the variables in the sampled data. However, modelling the variogram and kriging are complex to follow in order to achieve desirable results, as in geostatistical studies experts are required to perform their specialists’ tasks. To define this dependency, Cressie (1993) identified those tasks that an expert geostatistician and/or GI scientists would be required to perform to provide their analysis and prediction:

- Design the sampling plan;
- graph and summarise the data;
- detect and allow for spatial non-stationarity;
- estimate spatial relationships, usually through the variogram (structured analysis);
- estimate the *in situ* resources, usually through kriging; then
- assess the recoverable reserves and, if a decision is made to mine, determine the mine plan and provide current reserve assessments as the mining proceeds.

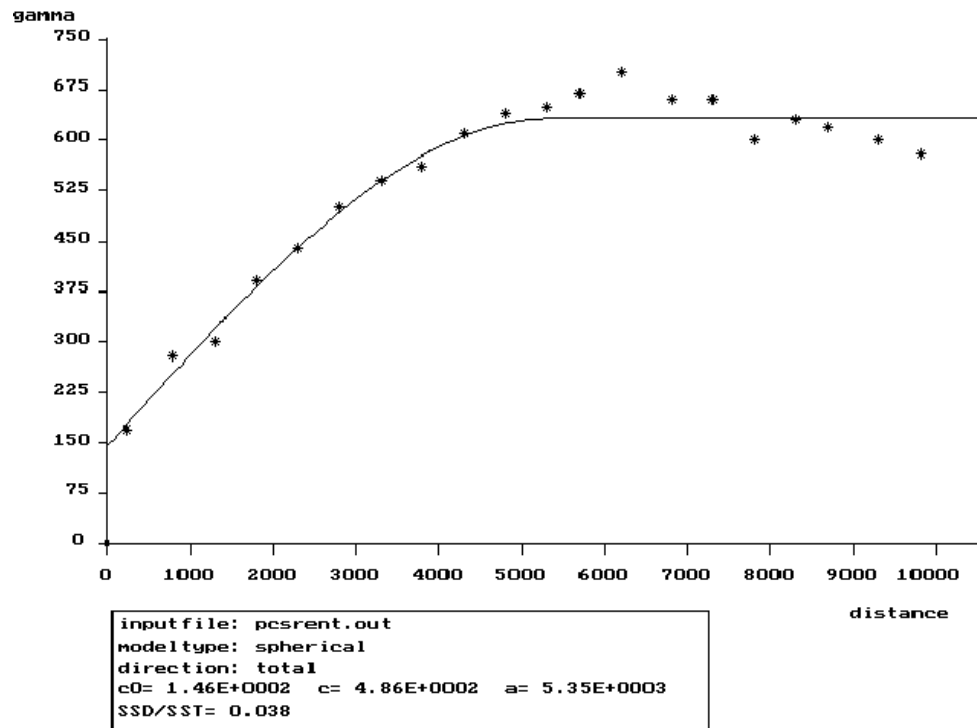


Figure 1.1: Example of a Variogram

Carrying out these steps in a coherent manner is an expert task. Currently, the only way to model and evaluate a variogram is by employing an expert who physically visualises the data, makes decisions regarding outliers, clusters and trend, and decides which mathematical functions to incorporate in order to obtain the characteristics of the subject in question. However, this thesis puts forward the possibility of allowing a variogram to be produced without necessarily relying on the human expert, but rather by utilising an expert agent-based module of a distributed component system. Considerable improvement for such modelling could be achieved by automation and the use of negotiation mechanisms. Software agents have a strong focus in their ability to negotiate and collaborate in order to produce a possible software solution. Software agents can be employed to provide the potential to solve the persistent problems associated with GIS (Goodchild, 2004; Reitsma, 2005; Brown *et al.*, 2005; Albrecht, 2005), such as those arising from geographical scale, as discussed by Atkinson and Tate (2000). Also, with the rise of geocomputation

(Openshaw and Abrahart, 2000) and ubiquitous computing (and consequently ubiquitous GIS, Longley and Batty, 2003), agent technology has been considered as potentially playing an important role, especially to solve issues associated with ontology for interoperability (Smith and Mark, 1998; Fonseca and Egenhofer, 1999), data and tool sharing and coupling (O'Reilly, 2005; Graham, 2005), and better knowledge dissemination among experts (and even students) (Frank and Raubal, 2001). This thesis explores this argument by investigate software agents applied to geostatistics, focusing on producing a variogram agent component (VAC) as proof of concept for intelligent distributed component GIS. This variogram agent component is conceptualised as a multi-agent system in which individual agents collaborate to achieve the goal of modelling the variogram. This software agent solution will aim to provide effective distributed components for GIS, and will be tested using a case study of known (published) data.

Software agents are already being utilised in GIS for the purpose of spatial simulation., A large part of these applications emphasizes individual entities, where each entity within a dataset is defined as an agent, to provide emergent patterns and relations that are formed due to the individual's interactions (e.g. Batty *et al.*, 2003; Batty, 2005a; Albrecht, 2005). Software agents have also been used to represent spatial processes (Reitsma and Albrecht, 2005). However, software agents can offer many more benefits to GIS. This could be achieved by having software agents not just acting as entities, but representing a software development paradigm in which the software could be fully developed using the syntax and semantics it provides, like other functional software systems that have been developed using software agent paradigms (Wooldridge, 2003; Bresciani *et al.*, 2004).

This research, therefore, aims to utilise a novel approach of using software agents to act as a tool to provide a service within GIS and to support the idea of distributed components GIS. As a way of investigating this, a distributed component – the variogram agent component, based on agents, will be developed. This is a challenging

task within geostatistics because each data set needs its own structure and procedure to produce an effective and useful variogram. The research will be based on the concept reducing reliance on expert interaction by letting the agents learn, communicate and collaborate in order to do what the human expert would have done. To examine the possibilities of improvement with the introduction of software agents, a modular approach to GIS tool development will need to be initiated, starting with the variogram agent as a key proof of concept.

1.3 Aims and Objectives

The aims of this research are to investigate software agent characteristics and environment architectures, and the current use of software agents in geographical information sciences, so as to understand the agent capabilities being utilised in GIS, model newer and more robust ways to try to increase agent functionality to their potential capacity, develop distributed component GIS, and test the newly developed service to justify the agent characteristics being focused upon and their functionality in distributed components GIS services.

Such knowledge would be useful for providing services and reducing the complexity of using GIS tools which will, in turn, reduce the expert user interaction and allow a more novice user to be able to comfortably interact with GIS services. The vision is to have GIS functions being used like those of mobile telephone functions (such as with SMS, where the user knows how to write the text, but not necessarily what the underlying technologies are that provide the transportation of that text to the intended recipient). Thus, the outcome of the research is to provide an architecture, platform and sample tool that would allow GIS queries and services seamlessly, for novice and expert users alike.

To pilot and set a structure for the thesis, objectives were determined as to what would need to be conducted for the research to be successful. The main objectives are to

identify the flow of events that produce the most effective variogram, the right variogram formulae to be supplied to the agents, and the right type of agent architecture that would be most effective for agent technology-based GIS development. The objectives of the study are to:

1. identify the appropriate theoretical perspectives for analysing, developing and testing an algorithm model for a software agent-based GIS component;
2. examine existing agent environment architectures and determine which is the best fit for the purpose of using as a variogram agent. Thus, to establish the limitations and produce suitable architecture (or the features to add to the architecture) to allow for easy, flexible and efficient GIS agents;
3. identify appropriate functions for a variogram agent;
4. implement the features identified in objectives 2 and 3 to the environment architecture identified in objective 1, then test them accordingly as variogram agents;
5. test the variogram agent system and analyse its performance; and
6. draw conclusions as to the feasibility and practicality of using agent software for distributed component services in GIS.

1.4 Structure of the Thesis

This thesis commences with Chapter One introducing the study area, outlining the aims of the investigation and the objectives of the thesis, defining the problem and outlining how it is expected to be solved. Here, the definition surrounding the problems faced in the physical and socio-economic environments which are interrelational and spatial will be observed, together with the available tools used to deal with them. The current issues in distributed component technologies, which include the use of agent-based software development and the limitations of the current architecture of commercial GIS software to help deal with interoperability and tool coupling for numerical simulation modelling using spatial dependency, geostatistics, variograms

and the reliance on the expert's use of kriging, and finally the possibility of intelligent distributed component tools combining agents and GIS, will be discussed.

Chapter Two covers the existing theory behind GIS, showing the importance of making it a real distributed environment. This section will show the interconnection between GI Science, GI Engineering and GI Systems, along with some key trends such as geocomputation with its relation to spatial data mining, agents and geosimulation. Issues of interoperability and tool coupling for the ubiquitous GIS and Web 2.0 will provide the case study of an agent-based intelligent distributed component GIS.

Chapter Three will concentrate on the theory of software agents and agent-based technologies by providing a definition and their historical development, characteristics and typical applications. It will then provide a more detailed discussion of the use of agents in GIS and their functionality in geosimulations, with its provenance and technological heterogeneity. It will then classify agent-based applications in the spatial domain through a discussion about confusion of object-entity, process and service in GIS. This chapter will also provide the notion of the variogram agent as a test-bed for IDC GIS.

Chapter Four discusses geostatistics and the function, application and trends of using the variogram. The different aspects of geostatistics will be discussed, including spatial autocorrelation, autoregressive models, (semi)variogram models and kriging. The importance of data aspects when modelling the variogram will be defined, which involves data cleaning, trend detection and removal, and cluster detection and sampling. Finally, cross validation and goodness of fit will be considered.

Chapter Five will expand upon agent development methodologies, with their limitations and required improvements. Issues of agent programming languages (and their structure) and platforms suitable for GIS agent technology will be discussed. The datasets for testing and experiment design will then be provided.

Chapter Six will discuss the problems of agent system development and deployment. A suitable structure of agent software development to influence GIS tools will be established and proven to be fit for purpose. The actual design (software) of the agent-based variogram model with its variogram agent (multi-agent) and sub-agents with their functionality will then be explained and tested for validity (internal and external).

Chapter Seven presents the experimental results, along with verification and validation, the problems encountered and fixes, performance testing, statistics, outputs and limitations.

The overall results from the experiment and the general outcomes of the thesis will be evaluated and discussed in Chapter Eight. Thus, this chapter will focus on the learnability of the agents and other important characteristics for geostatistics and eventually GIS. A review of the findings, comparisons to published literature and original contribution to knowledge will be provided. Finally, the overall conclusion and future research agenda will be presented.

CHAPTER TWO: INTELLIGENT DISTRIBUTED COMPONENT GIS

2.1 Introduction

In recent years there have been major developments in areas of computing such as the tightly coupled distributed engines, web 2.0 and the push towards web 3.0 on the Internet (Graham, 2005; O'Reilly, 2006; Singer, 2009) and the advantage that could be gained using thin clients for ubiquitous computing (Ki-Joune and Li, 2007; Poslad, 2009). With the strong market penetration of mobile devices and the ever improving mobile technologies (software and hardware), the market provides a good opportunity for GIS development especially for research. Furthermore, the functionality and software development opportunities using Open Source can provide a platform to GIS evolution. In this regard GIScience researchers are already taking advantage of these state of the art technologies with the creation of new GIS areas like OpenGIS. However, in some of these areas where GIS might benefit from the technology, some hindering problems exist. For example in ubiquitous computing, due to the system requirements and the difference it represents in terms of system platform offered by different providers, interoperability and tool coupling is of major concern. Thus an understanding of technologies such as CORBA, OLE, COM and ODBC becomes a must for creating the possibility of achieving the potential development. The opportunities for mashups of these technologies to improve GIS could be a framework, for example, to aid environmental modelling especially with the aid of artificial (AI) technologies like the neural networks, fuzzy logics, pattern recognition and for the main interest of this research the software agents. Furthermore this would lead to tool coupling, a concept that is high on the agenda of GIS community (Fonseca *et. al.*, 2000; Albrecht, 2007, Torrens, 2007b) as is anticipated to improve data analysis.

Tool coupling can be enhanced by software agents with their ability for being efficiently distributed. This brings us to the core studies of this thesis which is to look into the possibilities of distributed component GIS through the use of software agents and their capabilities. The technology behind the tool coupling is currently being initiated into web maps, geomarkup language (GML) and many other GIS related areas. The research in this thesis demonstrates the important role software agent will play in distributed GIS and from this the development of intelligent distributed component GIS. This can be achieved with the help of currently emerging technologies in GIS like spatial data mining tools and artificial intelligence contributing to the field and which have led to technologies like the Geosimulation which is developed through the advancement of geocomputational theories (Batty, 2000b; Albreicht 2005; Batty, 2005a).

The focus for this chapter is to develop a justification, through the literature, for the use of agent technologies to facilitate intelligent distributed component GIS. Chapter 3 will then focus in-depth on agent technologies and their current use in GIScience.

2.2 GISystems, Science and Engineering

Since their origins in the mid-1960s, GISystems have been well established for storing, handling, analysing, visualising and disseminating spatial data. As a technology, GISystems have become successfully integrated with other technologies from different disciplines, such as geography, computer science, environmental science and demography. They are also used in a wide range of areas, including environmental protection, construction, transportation, education, crime, health and planning.

“[GIS is] ...a collection of computer software tools that facilitate, through georeferencing, the integration of spatial, non-spatial, qualitative and

quantitative data into a database that can be managed under one system environment" (Cressie, 1996: p118).

However, Wright *et al.* (1997) associate GISystems as being a tool that could help to solve spatial problems, or simply help to build tools that could be added to existing GISystems to help improve solutions to spatial problems, or simply to study and understand the theory and concepts that lie behind the tools.

By the early 1990s, the recognition of a coherent GIScience was beginning to be debated (Goodchild, 1990, 1992). A range of research, relating to different aspects, can be classified into GIScience. Some of these are within the context of many other research disciplines, as areas related to ontology, representation, computation, cognition, uncertainty, visualisation, institution and society, all of which also constitute GIScience research (Mark, 2003). Although GIScience is not that neatly summarised, as the questions and issues are either solved or better understood so the GIScience research agenda evolve and continually moves on (Brimicombe and Li, 2009). Three main elements are identified by Goodchild (1992) to make GIScience a separate domain of research discipline:

- the use of the spatial key $\{x, y, a_1, a_2, \dots a_n\}$, where $\{x, y\}$ defines location as continuous dimensions and $\{a_1, a_2, \dots a_n\}$ define the attributes of location either as continuous or discrete dimensions;
- the presence of spatial dependence between locations, in that near things are more likely to be similar than distant things;
- the durability of the spatial data primitives of point, line, polygon and cell/pixel that have underwritten the technology and its application in many diverse applications.

Spatial dependency is a key issue in GIScience. It derives from Tobler's first law of geography

“...everything is related to everything else, but near things are more related than distant things” (1969: p7).

It provides a basis for which spatial phenomenon can be studied and understood. Considering the importance of spatial dependency, the means to analyse this is regarded as an essential part of GIScience (Wright *et al.*, 1997). In this research, spatial statistics (commonly known as geostatistics) is referred to as one approach for dealing with this issue in GIScience.

Goodchild (1997) further expresses GIScience as being comprised of information about places on the Earth's surface, knowledge about where something is and knowledge about what is at a given location. GIScience is the field of study that addresses the production of geographic data, the transformation of data into useful geographic information and the construction of geographic knowledge. This relates both to physical and biological aspects of the environment (including flora and fauna) and the social, cultural and economic aspects of the lived environment.

The GIScience concept encapsulates both reductionist and holistic approaches to the study of geographical phenomena. To study these complex issues, researchers have tended to develop tools that simplify reality as models and be able to handle the large amount of information produced. GISystems are, as such, collections of tools for the resource intense GIScience. Ball and Babbage (1989) acknowledge that GISystems assist GIScience studies in two important factors: human and physical. Thus, GISystems and GIScience could be regarded as integral, symbiotic parts of the same domain or, at the same time, could be differentiated as GISystems being the front end tool that allows spatial

analysis, with GIScience providing the underlying theory that supports the use of this tool. This view has been widely discussed and generally accepted (Goodchild, 1992; Goodchild, 1997; Wright *et al.*, 1997; Mark, 2003). Therefore, in this thesis, the term GIS will be used to signify both GISystems and GIScience

There is an important distinction to be made between science and the process of engineering the system to study its phenomenon. Frank and Raubal (2001) define and differentiate science to engineering as:

“Science is the search for knowledge (new knowledge, to be precise), and engineering is the systematic application of the results of scientific research to solve real-world problems in a predictably successful way” (Frank and Raubal 2001: p12).

Here we can see a new concept emerging; that of GI-Engineering. This concept interfaces and connects both GIScience and GISystems. Frank and Raubal (2001) state that “GI engineering is the scientific efforts to establish the rules and heuristics which engineers can use to build GI systems that predictably work”, while enforcing the point as to the importance of GI-Engineering by stating:

“...the scientific results that we have produced over the past decades are substantial and cover most aspects of GI. However, they are not yet ‘reduced to practice’ to be usable to design systems that predictably work” (Frank and Raubal, 2001: p13).

GI-Engineering has been defined as:

“the design of dependably engineered solutions to society’s use of geographical information” (Brimicombe and Li, 2009: p103).

These solutions are naturally developed on GISystems and GIScience and may also employ technologies from other disciplines. GI-Engineering aims to design products that integrate various technologies and techniques seamlessly so they are easy to use. Any specialist knowledge of GIScience and GISystems should not be required to use these

products. By bringing GISystems and GIScience together, GI-Engineering is expected to help take GIS into mainstream IT and become widespread within society. GI-Engineering is an important aspect of this thesis.

2.3 Some Key GIS Research Themes

A number of key developments in GIS research underscore the aim of this thesis to investigate intelligent distributed component GIS. These can be categorised under the following headings:

- GIS and environmental modelling
- Spatial data mining
- Geocomputation and geosimulation
- Web 2.0 (current a concept persist for web 3.0 which is to be based on semantic web)
- Ubiquitous GIS

2.3.1 GIS and Environmental Modelling

To assess the many factors affecting environmental change, the use of environmental modelling can often be effectively employed, as it is able to provide predictive and diagnostic outputs which can be used to measure the impact on the environment, allow efficient management of natural resources and plan for contingencies. It assesses most aspects of nature, including the quality of air, soil and water, waste products, biological factors, hydrology and noise pollution. Environmental modelling can be considered a part of the spatial domain.

There are three broad approaches to environmental modelling: conceptual, numerical and physical (analogy). However, the only models to be considered further here will be numerical models that require computers for their execution.

Environmental models usually contain a degree of spatiality, though one-dimensional process models do not explicitly express the spatial dimension. Even in two-dimensional and three-dimensional process models, the governing equations may be unable to fully express the spatial dimension. Often due to a lack of software development, some spatial components may lack flexibility or are inaccessible. However, GIS are already capable of addressing such spatial information, which could assist such models in successfully handling spatial data. To this end, the integration of such models with GIS is becoming accepted as being useful even *de facto* when handling environmental problems (Brimicombe, 2009).

GIS and numerical environmental models can be coupled in a number of ways (Karimi and Houston, 1996; van Vliet *et. al.*, 2009). This can be from loosely-coupled (whereby the only interaction which occurs is the exporting or importing of data via an exchange format), through to tightly-coupled (allowing a high degree of interoperability, such as Windows compliancy), and tool or network coupling where from a single interface the user can access distributed data, model subsystems and computing resources seamlessly across networks. For simpler modes of coupling, the option then is whether to develop the required tools externally of GIS or within it. A number of issues have to be considered in this respect, such as the limited capability of most GIS to store imprecise or spatio-temporal data; limitations in functionality for most GIS regarding spatial data analysis (such as in geostatistics), how easy it is to share data, the running speed of the model and flexibility in the choice of the model. Tool or network coupling opens up the prospect of

integrated GIS and numerical environmental modelling using distributed components incorporating agent-based technology to reduce the level of complexity to the user.

Considerable proprietary (e.g. ArcGIS) and public domain software (e.g. GeoDa) exists which can be effectively utilised for analysing spatial data. As such, in regard to environmental modelling and considering the increasing sophistication, speed and interoperability of such software, there is fast becoming little reason to spend money re-implementing such tools within GIS and every reason to assemble them outside of GIS (Li *et al.*, 2000). This concept can also be extended toward other applications where it is both feasible and flexible for GIS to be linked, via an interoperable framework, with external numerical models.

2.3.2 Spatial Data Mining

Spatial data mining is the process of having spatial information and knowledge extracted from a database or data warehouse, containing both spatial and attribute data, where the data warehouse is a repository of a large amount of data brought together from multiple different sources (Connolly and Begg, 2004). A data warehouse mainly contains previously used data which may have different data structures and can be used for a range of analyses. It thus brings with it certain difficulties to efficiently extract useful information from such a data warehouse. Sometimes, data mining is also termed 'knowledge discovery in databases' (KDD) (Ding, 2007). Another definition of the process for KDD is:

“...interactive and iterative, involving several steps such as data selection, data reduction, data mining, and the evaluation of the data mining results” (Fayyad *et al.*, 1996: p83-84).

The heart of the process, however, is the data mining step which consists of the application of data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data (Ester *et al.*, 2001).

Since the term ‘spatial data mining’ comes from a combination of two well revised components (spatial; data mining), it is important to understand the need and function of the term as a whole in the designated definition. When referring to KDD (in a generic sense), data mining is explained by Ester *et al.* (2001) as:

“[a] non-trivial process of discovering valid, novel, potentially useful and ultimately understandable patterns from data, a pattern is an expression in some language describing a subset of the data or a model applicable to that subset” (2001: p3).

KDD is therefore based on the premise that interesting patterns are hidden in and can be extracted from very large databases (Miller and Han, 2009). Meanwhile, the term ‘spatial’ introduces issues within the data of geographical relations. As Shekhar *et al.* (2003) point out, the difference between non-spatial and spatial objects is that:

“Non-spatial objects are explicit in data inputs, e.g., arithmetic relation, ordering, is instance of, subclass of, and membership of. In contrast, relationships among spatial objects are often implicit, such as overlap, intersect, and behind... Extracting interesting and useful patterns from spatial datasets is more difficult than extracting the corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation” (2003: p64).

Following the above discussion, Table 2.1 shows the distinction between spatial and non-spatial data mining. Prasad *et al.* (2007) further explain that: “The explicit location and extension of spatial objects define implicit relations of spatial neighbourhood (such as topological, distance and direction relations) which are used by spatial data mining algorithms” (2007: p1).

Table 2.1: Search Patterns for Classic Data Mining and Spatial Data Mining

	Classical Data Mining	Spatial Data Mining
Predictive Model	Classification accuracy	Spatial accuracy
Cluster	Low coupling and high cohesion in feature space	Spatial continuity, unusual density, boundary
Outlier	Different from population or neighbours in features	Significant attribute discontinuity in geographical features
Association	Subset prevalence, $Pr[B \in T A \in T, T: a \text{ transaction}]$ Correlation	Spatial pattern prevalence, $Pr[B \in N(A) N: neighborhood]$ Cross K-Function

Source: improved from Prasad *et al.* (2007).

There has been an explosive growth in geographical data since the mid-1990s due to a combination of GPS, remote sensing and cheaper, easier data storage. However, conventional spatial data mining has been predicated on powerful, centralised computing and algorithms that search for interesting patterns. Spatial objects and relations tend to be more complex than those found in non-geographical databases leading to computational complexity. Fortunately, many spatial analytic techniques can be decomposed into parallel and distributed computations (Miller and Han, 2009) across networks. Newly emerging is distributed data mining which makes use of networked environments to mine distributed and heterogeneous data without the need to collect them into a centralised repository (Datta *et al.*, 2006; Laube and Duckham, 2009). This has naturally required the use of agent technologies (Laube and Duckham, 2009) in order to implement algorithms that can act in a decentralised way to effect knowledge discovery.

Brimicombe regards

“With the exponential rise in the size of databases/data warehouses, their increasingly complex structures and the rate at which they can accumulate data on even a daily basis, there is an urgent need for techniques that can mine very large databases for the knowledge they contain. Various spatial data mining techniques have thus been developed for the discovery of meaningful patterns from large datasets, where an important dimension of interest is the geographical location of events” (2003b: p2).

2.3.3 Geocomputation and Geosimulation

Geocomputation (Longley *et al.*, 1998; Openshaw and Abrahart, 2000) was a paradigm shift that came in the late 1990s with the maturation of GIS and the rise of research within a computational environment:

“Geocomputation is a follow on revolution that is occurring after the introduction of geographic information science (GIS). It is expected to gather speed and momentum in the first decade of the 21st century... when we have finished creating our GIS databases, set up our digital spatial libraries, and expanded to include everything that can be linked into a two- or three-dimensional geographical coordinates then we are all set for [Geocomputation]” (Openshaw and Abrahart, 2000:13).

The main challenge is how to make better and fuller use of this stored spatial data. In this regard, geocomputation can be considered to be the utilisation of spatial computation tools and methodologies in order to solve applied problems. Such tools and methodologies would automatically include the use of GIS, although it must be acknowledged that the present functionality of commercially available packages is often inadequate, barely extending beyond the routine handling and analysis of spatial data (Gahegin, 1999). As a result, other tools and methodologies are being sought and used, either on their own or combined with GIS, including AI, neural nets, fuzzy computations and genetic algorithms (Li *et al.*, 2000, Agkun, *et. al*, 2008, Soleimani, 2009). This was a conscious attempt to enrich the available spatial analysis toolkit, provide greater flexibility in moving away from just monolithic GIS on its own and has inevitably lead to the technological heterogeneity of tool coupling.

As there are increases in both the size of datasets and complexity of spatial analysis, so a conventional approach to GIS becomes inadequate (McMaster and User, 2004), particularly reliance on single CPU whereas distributed computing is required for the computational intensity needed. As a new paradigm, there are many debates around issues such as 'what is the exact definition of the term?', 'what should it be encompassing?', and many other technological questions. The extreme of this debate questions whether it will make any real contribution to the sciences:

"Geocomputation is not just using computational techniques to solve spatial problems, but rather a completely new way of doing science in a geographical context" (Openshaw, 2000: p.3).

But as Kirkby notes:

"As with other aspects of research, there is a need to use computing tools with discrimination, and not assume the power conquers all" (2000: p7).

Nevertheless, one distinct advantage of geocomputation is in regard to the experimental and creative use of GIS for interaction, dynamics and process, rather than passive responses, statistics and form (Longley, 1998). For technologically heterogeneous applications (such as numerical simulation modelling), geocomputation can be regarded as being central to the way GIS are used (Brimicombe, 2003).

Geosimulation can be viewed as an extension to geocomputation, being distinguished from other simulation methodologies due to its particular and explicit attention to geography and space (Benenson and Torrens, 2004a). Some interesting research has been undertaken recently which, although spread over many different areas, can also be categorised into geosimulation (Albrecht, 2005). One example consists of the development of a series of relatively simple decentralised, individual-based models

allowing contemporary theory of urban planning to be tested using traditional GIS data (Benenson and Torrens, 2004b; Torrens, 2007a). By bridging the process/object boundary, new data representations have been studied which capture the essence of geodynamics (McIntosh and Yuan, 2005). Software packages have been developed using interoperable environments and multiple agents, which provide a practical approach to surmounting the complex problems of geographical systems (Waddell *et al.*, 1993; Brown *et al.*, 2005).

Geosimulation is about modelling complexity and emergence (Batty, 2005b) and in GIScience has become a key focus for experimentation and theory building/testing *in silico* (Brimicombe *et al.*, 2009). Central to geosimulation is computation and the use of agent technologies (further discussed in Section 3.3). The level of complexity that can be reached using agent technologies poses methodological challenges in establishing and validating plausible models (Gilbert and Banks, 2002; Manson, 2007). To overcome some of these difficulties, Li *et al.* (2008) have proposed utilising agent-based technologies to create services that help to calibrate and validate multi-agent models, which have been implemented within a multi-agent way-finding model (Li *et al.*, 2008). In conclusion, it is worth pointing out that current research indicates that agent technology has considerable potential in geosimulation.

2.3.4 Web 2.0

It has been nearly a decade since the start of Web 2.0. However, even the term 'Web 2.0' itself has proved controversy, with some people declaring it to be a meaningless marketing 'buzzword', but others accepting it as a new technological breakthrough. It aims to provide functionality to the 'always online' websites and web applications, so that they become interlinked and act as computing platforms with applications which appear to be available

offline to the user who is able to generate and distribute content at any time (Barnwal, 2007).

The term 'Web 2.0' was coined during the first O'Reilly Media Conference in 2004 (O'Reilly, 2005; Graham, 2005). The main aim of the concept is not to upgrade the web engine, but to change the way software developers and end-users utilise the Internet, which is to encourage creativity through communication (information sharing and collaboration).

"Web 2.0 is the business revolution in the computer industry caused by the move to the Internet as platform, and an attempt to understand the rules for success on that new platform" (O'Reilly, 2006: p1). Best (2006: p7) describes the Web 2.0 characteristics as "rich user experience, user participation, dynamic content, metadata, web standards and scalability", while Greenmeier and Gaudin (2007) add characteristics like openness, freedom and collective intelligence by way of user participation (O'Reilly, 2005). Currently, there are various definitions of Web 2.0, for example:

"The philosophy of mutually maximizing collective intelligence and adds value for each participant by formalized and dynamic information sharing and creation" (Högg *et al.*, 2006: p13).

"All those Internet utilities and services sustained in a data base which can be modified by users whether in its content (adding, changing or deleting-information or associating metadata with the existing information), or how to display them, or in content and external aspect simultaneously" (Ribes, 2007<http://www.campusred.net/TELOS/articuloperspectiva.asp?idarticulo=2&rev=73>)

It is important to be aware that the term '2.0' does not refer to the second version of the Web system or the www (Anderson, 2006; Berners-Lee, 2006). According to Decrem (2007), it is the 'participatory web', on which the information sources are exactly the same as the Web version that everybody is familiar with.

The purpose of the perceived infrastructure is to allow the interactive facilities of the 'normal web' to be redefined as a platform that allows users to run applications entirely through a web browser and allows them to own data with an option to exercise control over that data (O'Reilly, 2005; Hinchcliffe, 2006). The application should (or rather will) also act as an API that provides an opportunity to the user to add value to them (Graham, 2005). O'Reilly (2005) defines this concept as 'architecture of participation'. This then becomes a web architecture that allows users not only to retrieve information but also to contribute to the application and information.

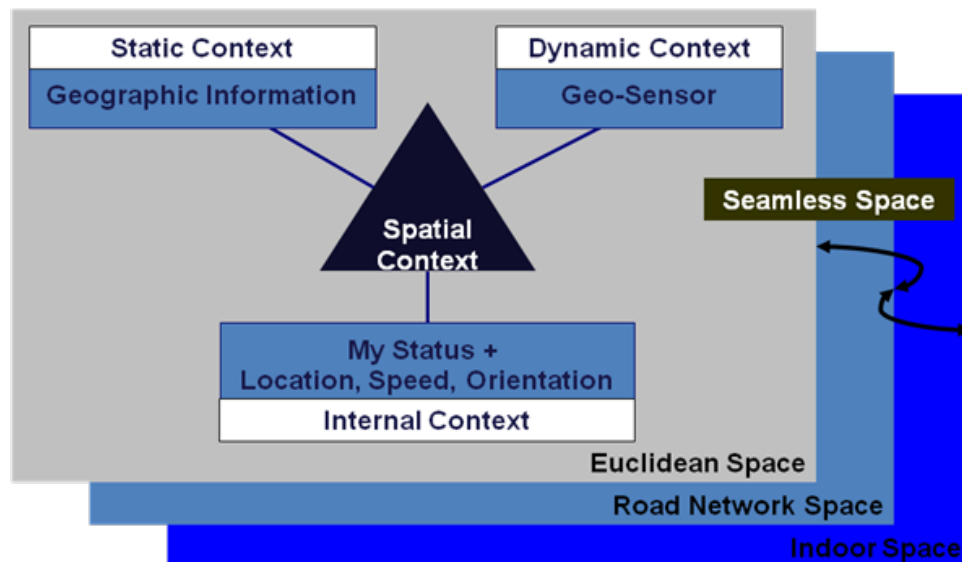
Web 2.0 includes features like the Mashups which are centred on merging content from different sources (client- and server-side). The notion of 'Mashup' was first derived from the concept of having an open API that allows a user to modify, change or add into the software, so that a completely new service is created. WebMashup (2007) further explains that "content used in Mashups is typically sourced from a third party via a public interface or so called API". The Google Maps Mashup is Google Maps embedded onto web pages of outside developers through the Google Maps API, using a simple JavaScript interface or a Flash interface. The Google Maps API includes language localisation and geocoding, and has mechanisms for enterprise developers who want to utilise the Google Maps API within an intranet. MapTube is a website for sharing maps, which also intends to follow the spirit of Web 2.0 (CASA website, <http://www.maptube.org/casa.aspx> accessed 12/03/2009). The maps themselves are not stored on the server, but link to another website where the map is already published. When maps are shared, information about what the map is and what it shows is entered by the owner and this is stored on the server along with the link to where the map is published. Implications of Web 2.0 for GIScience are crowd-sourced data and the delivery of distributed spatial information to clients, particularly mobile devices having limited local computing power.

2.3.5 Ubiquitous GIS

Currently mobile communication devices are considered as multipurpose computers, like desktops and laptops which is a phenomenon that has been observed at least before year 2005 (Mupparapu *et al.*, 2005), providing a sense of ubiquitous computing. Nowadays, mobile devices come with extensive power in terms of processing mechanism, random access memory and hard drive/flash memory. As the devices, networks and operating systems are becoming ever more sophisticated; they increase the span of mobile users and their ability to explore the limitations of the devices. Communication technologies, from the forerunner of fixed landlines through to mobiles, have provided a completely new range of wearable technologies. Currently, mobile usage ranges from being a simple means of communication to a seriously powerful data storage device with a capability similar to a contemporary computer. Many in the business industry are already using mobile devices as mobile offices and determined to be functional for e-mail exchange and data access, which can undertake downloads and uploads of data.

With massive decentralisation of computing, data and software are being distributed across networks, with access increasingly being affected by mobile devices (Longley and Batty, 2003). The rapid development of Information Communication Technologies (ICTs) and the remarkable convergence between separate disciplines have led to the diversity of GIS applications, and indicate the potential for GIS to become ubiquitous as mobile geographic services. Zipf and Jöst (2006) discussed the technical and usability issues related to the combination of GI services and ubiquitous computing, which they term 'UbiGIS', and there is also research being undertaken on location base services (LBS) and spatial knowledge acquisition which is also based upon ubiquitous GIS (Brimicombe and Li, 2009).

Other relevant applications and areas of discussion include context-awareness (Jiang and Yao, 2005), LBS (Schmidt *et al.*, 1999), the importance of UbiGIS (Reuter and Zipf, 2005), environmental monitoring by new mini-sensors (known as Smart Dust), and telematics and logistics (Zipf and Jöst, 2006). Zipf and Jöst (2006) have also reported on the heterogeneity and current expansion into computing, and the improvements in strength and capability of mobile devices. Li (2006) has developed a structure for location awareness in ubiquitous computing by providing spatial context-awareness (see Figure 2.1).



Source: Li (2006)

Figure 2.1: The Ubiquitous Computing Environment for GIS

Section 2.3 has focused on some trends for GI research which reinforce the point that there are drivers for distributed component GIS and for GIS to be integrated into technologically heterogeneous environments. Interoperability and tool coupling will now be discussed in the next section.

2.4 Interoperability and Tool Coupling

2.4.1 External Coupling of GIS

Section 2.3.1 introduced the strategies of coupling environmental modelling with GIS. For similar reasons, it would be useful to couple GIS with external tools and external numerical simulation models (Brimicombe, 2003). Such external models and tools have their own purposes and strengths, but they cannot always fully represent the spatial dimension. Using such other tools to model the spatial component often lack sufficient flexibility or it is not possible. On the other hand, although GIS is well developed to handle spatial information, it is not particularly well designed to incorporate a mechanism enabling dynamic simulation, complex data management, statistical analyses and numerical computing. GIS functionalities may be enhanced through the borrowing of technical capabilities from simulation models and external tools. The current trend for ubiquitous computing, as well as the growth of information communication technology, supports such coupling strategies.

Fully embedded models and complex tools within GIS software packages are not considered to be the best option, due to their limited computation speeds, functionality of spatial data analysis and data management capability (for example, DBMS such as Oracle have the strength to store large amounts of data and can incorporate a comprehensive query function). However, it is not yet possible to build such DBMS into GIS software, although coupling GIS with external DBMS does occur. Statistical tools are often externally coupled with GIS software, and there are various tools which can provide the powerful statistical analysis capability required for GIS projects. Embedding such statistical functions within GIS software would be costly and unwieldy, but external links with GIS can be both efficient and effective. For some specialist software tools, external coupling with a GIS is both robust and straightforward - such as for neural nets, where it is neither

necessary nor pragmatic to develop complex model building tools within the GIS software. However, external to GIS, neural nets can assist many GIS applications, such as road network management or land use modelling.

If we consider the case of hydrology, where a hydrological model normally requires elements of the drainage basin to be stored as reaches, junctions and sub-basins. Here, a GIS model could simplify matters, and coupling the available external hydrological models to such GIS software would improve outputs. Similarly, ecological models focus on the dynamics of simulation. A wide range of data models exists within ecological modelling. Each model needs to address the numerical methods required to solve the particular process under simulation. Coupling relevant GIS software with these external ecological models could offer an effective solution, as currently GIS software is weak for temporal data models. GIS software can be joined with external socio-economic models as well as environmental models. Considering the development of a simulation for urban sprawl, models could be devised with very different technologies (such as agents or fractal), and the coupling of GIS software with external models is indeed generally accepted within such studies. However, needless to say, it is neither realistic nor necessary to develop all of these models within GIS software packages.

2.4.2 Interoperability

Albrecht (1996) defines interoperability as the ability of client-side software applications to access a service from a server-side implementation such that it will respond as expected. Interoperability, in GIS terms, can be described as:

“...openness in the software industry, because open publication of internal data structures allows GIS users to build applications that integrate software components from different developers, and it allows new vendors to enter the market with competing products that are interchangeable with existing

components, just as the concept of interchangeable parts helps competition in the automobile industry” (Goodchild *et al.*, 1997: p2).

Brimicombe (2003) further clarifies that interoperability brings together software at a more structured developmental level than is usually found through the integration of independently developed software. This involves software parts that are interchangeable so that, within a specific hardware and operating system environment, groups of software can seamlessly operate together. Interoperability not only focuses on sharing interfaces and data, it also involves sharing functionality. In other words, interoperability is the concept of having multiple forms of tools and data, and being able to mix them together to achieve a common goal.

Market dominant operating systems (such as Microsoft Windows and Unix), together with object-oriented programming (OOP) and standardised Internet protocols, have contributed to an environment of mutual interoperability. This brings three important advantages. The first is that most software tends to have the same look and feel, a shared principle of interface design. Secondly, data access over networks is eased as communication protocols and data services have become largely transparent to the users. Thirdly, with OOP high level languages, it is possible to select services from more than one currently available software and wrap them in a common interface.

The Open Geospatial Consortium (OGC, previously known as the OpenGIS Consortium) was established in the mid-1990s as a way of fostering interoperability. OGC aims to produce geospatial data and geoprocessing resources that are fully integrated into mainstream computing, and progress toward the widespread use of interoperable, commercial geoprocessing software throughout the global information infrastructure (McKee, 1996). Concepts of OpenGIS are supported by technologies such as the Internet, CORBA, OLE/COM and ODBC. The framework of OpenGIS specification includes:

- a common means for digitally representing the Earth and Earth phenomena, mathematically and conceptually;
- a common model for implementing services for access, management, manipulation, representation, and sharing of GeoData between information communities; and
- a framework for using the open GeoData model and the OpenGIS services model to solve not only the technical non-interoperability problem, but also the institutional non-interoperability problem (Buehler and McKee, 1998).

From the perspective of computer science, interoperability means the ability of one system to seamlessly access information from another system. As it allows data and other resources to be shared across a network, the full strength of a computer system would be harnessed. It could, therefore, enable the sharing and exchange of information in heterogeneous and distributed computing environments (Yuan, 2001). One solution to interoperability is to use autonomous entities (Wooldridge and Jennings, 2000; Brown *et al.*, 2002; Bigus and Bigus, 2003; Torrens, 2003; Wooldridge, 2003; Sherematonov *et al.*, 2004; Albrecht, 2005) and processes (Wooldridge and Jennings, 2000; Bigus and Bigus, 2003; Wooldridge, 2003). This solution aims to provide a cost effective and user friendly means to maximise the usefulness of independent information and computing resources across multiple platforms and institutions. Agents can act as such autonomous entities or processes. Agents may enhance interoperability, because they are platform independent and even program independent (in other words, one agent developed in C++ should be able to communicate another developed in Java). Strong research results (Li, 2006; Li *et al.*, 2008) demonstrate that interoperable multi-agent systems, which are developed over networks, are efficient and effective in GIS applications.

2.4.3 Tool Coupling

Broadly speaking, tool coupling is the integration of two or more distinct technologies, such as GIS, databases and numerical simulation models (Brimicombe, 2003). According to the typology developed by Brandmeyer and Karimi (2000), tool coupling is a modelling framework having sub-systems wrapped within a common user interface. The sub-systems could reside on the same computer, but could also be distributed over a network. Sub-systems can be specific to data management, spatial data manipulation and analysis, model building and management. Within the framework, these sub-systems can be coupled together for one common goal. Normally, it would be expensive to design and develop such a framework. The following examples show how the concept of tool coupling is implemented in various areas.

LandSerf was originally developed on the basis of Jo Wood's initial idea in his PhD thesis (Wood, 1996). Currently it is a free software package for the processing, visualising and analysing of surfaces, such as the Digital Elevation Models (DEMs). It is written in Java and runs on any platform that supports the Java Runtime Environment, such as Windows, MacOSX, Unix, Linux and Symbian (see Chapter Five for more details). Currently, its usage is widespread in geomorphology, ecology, archaeology, 3D gaming and GPS mapping. LandSerf couples a series of tools for multiple surface model handling, interactive 3D viewing, 'flythrough' of surface, lighting/shade models, multiple image blending, dynamic graphical query, raster and vector transformation, multi-scale surface processing and so on.

GeoDa. Following the trend in functionality and working concepts of interoperability, GeoDa is designed to implement techniques for exploratory spatial data analysis (ESDA) of lattice data (points and polygons) (Anselin *et al.*, 2006). It is a free software program

which provides a user-friendly and graphical interface for non-GIS specialists. It conducts descriptive spatial data analysis, geovisualisation, spatial autocorrelation and spatial modelling (Anselin, 2005). GeoDa is written in C++. It contains many features for spatial data manipulation, visualisation and query, mapping, exploratory data analysis and spatial statistics, which include global and local spatial autocorrelation, spatial regression, cartogram and 3D visualisation. A key feature of GeoDa is an interactive environment that combines maps with statistical graphics, using the technology of dynamically linked windows. It provides the user with multiple views for one project.

CrimeStat. This is a full-featured Windows-based spatial statistics program that provides statistical tools to aid law enforcement agencies and criminal justice researchers in their crime mapping efforts. It interfaces with most GIS software packages and is currently used by many police departments. It mainly focuses on the analysis of crime incident location. CrimeStat III is the latest version of CrimeStat. CrimeStat III couples different tools for distance analyses, hot spot analyses, interpolation, journey to crime analysis, as well as crime travel demand modelling. A number of different spatial data analysis techniques are employed in CrimeStat III, such as the kernel density technique for interpolation and the 'fuzzy model' for hot spot analysis. An example of the successful usage of CrimeStat III is reported by Levine (2007), who used it to model bank robbery trips.

Geographically weighted regression (GWR). This is a technique used to analyse spatially varying relationships (Fotheringham *et al.*, 2002), the software of which utilises three models (Gaussian, Logistic and Poisson) with a user-friendly interface. Models can be fitted to the spatial data under examination. A 'standard' Gaussian model is available if the response (or dependent) variable is able to sensibly take any value on the real line. Alternatively, should the response variables take the values of 0/1 (true/false) only, then a logistical model will provide location specific estimates of the probability of the response

variable being unity. Finally, where the data consist of positive integer counts, then a Poisson model would be appropriate. Outputs from this software are able to provide a comma-separated variable file for other statistical programs (such as SPSS), and also allow a convenient link to mapping software (such as MapInfo or ArcMap). GWR software provides a typical example of tool coupling in geostatistics.

2.4.4 Coupling GIS for Better Data Analysis

The external coupling strategy of GIS software which can enhance both the functionality of spatial data analysis and the capability of data management in current GIS software packages has been considered (Karimi and Huston 1996; Tait *et al.*, 2004; Tang, 2008; Bhatt *et al.*, 2008), as well as the increase in computation speeds and ease of data sharing (Croner *et al.*, 1996; Hengl *et al.*, 2009). This improves the flexibility of GIS software towards other modelling tools and data. Interoperability has been clarified as being the bringing together of independent software at a more structural, software developmental level in order to operate together more efficiently and achieve common goals (Marshall, 2002; Fonseca *et al.*, 2000; Vckovski *et al.*, 1999). A number of relevant issues were introduced, such as OOP, standardised Internet protocols, OpenGIS and agent technology. The tool coupling strategy was discussed and some typical examples provided to demonstrate how this concept can be implemented in the different applications of spatial data analysis.

There is considerable GIS and application research with regard to interoperability and tool coupling (Fonseca *et al.*, 2000; Weidmann and Girardin 2006). It reflects the increasing demands for robust and flexible GIS capabilities and functionalities for the transformation, visualisation, analysis and simulation of spatial data. Some advanced technologies (such as intelligent agents, mobile communications and the Internet) provide enormous potential

to enhance GIS capabilities and functionalities. Development based upon such advanced technologies will lead to intelligent distributed component GIS.

2.5 Intelligent, Distributed Component GIS

Over recent years, advanced information and communication technologies (ICTs) have developed in the areas of mobile devices, wireless communications and the Internet, including personal digital assistants, GRID, software agents, digital libraries and interoperable systems. ICTs make GIS more robust, powerful and flexible, being also better at communicating and sharing geographical knowledge. Through integration with ICTs, GIS is evolving as a networked-based geographic information service that is both open and distributed (Peng and Tsou, 2003; Goodchild *et al.* 2004; Dragicevic and Balram, 2004; Amirian and Mansurian, 2006; Garawski and Pluciennik, 2006). Geographic information, GIS solutions and spatial analytical tools are increasingly being accessed by both wireless and wired networks.

The transformation from separate and unconnected computers to networked terminals and from fixed equipment to mobile devices represents a new frontier for GIS (Rehrl *et al.* 2003; Batty, 2005). Consequently, new developments are based on distributed component frameworks instead of the client/server computer model (Stojanovi and Djordjevic-Kajan, 2001) - as such, they are all within the scope of distributed component GIS. Here, GIS data is disseminated across networks, while GIS functionalities are decentralised through distributed computing. Due to such progress, GIS has moved further towards becoming ubiquitous, as wireless and mobile geographic services are able to deliver data, computing capabilities and integrated functionalities to distributed locations across various networks (Li and Maguire, 2003). Examples of distributed component GIS include the Internet GIS,

wireless and mobile GIS, OGC standard, Web mapping, location-based services and geographic markup language (GML).

2.5.1 Rationale of Distributed GIS Components

Distributed component GIS can offer solutions to overcome the everyday difficulties or improve the effectiveness of many GIS applications. The major barrier preventing the employment of GIS techniques to integrate GIS and environment modelling is the diversity of environmental modelling (Brimicombe, 2003). The many and varied environmental simulation models may well have very different structures, algorithms and data formats, or the same model may have to be adjusted for use in different locations. There is often a lack of expertise for specific environmental modelling and GIS techniques, which are both needed for integration (Muetzelfeldt and Duckham, 2005). Model developers and users have grown to expect the availability of generally applicable toolkits for GIS techniques which can be utilised for environmental simulation modelling when integrated with GIS. Today's GIS software packages have limited functionalities to deal with spatial problems in environmental simulation modelling (Repar, 2005). Potentially, distributed GIS components could be developed to provide interoperable GIS tools and functionalities across networks which would couple with the various environmental models.

A number of disciplines have already utilised spatial data mining techniques (Okwangale and Ogao 2006). A variety of data will need to be mined to achieve very different objectives. These are thus the demands of having to handle diverse tools to solve spatial problems, which GIS software functionalities are currently unable to do, although considerable effort has been made to couple together the various spatial data analysis tools. However, distributed GIS components work in the opposite way. They are able to

efficiently disseminate spatial data mining techniques on-line, offering more robust off-shelf tools to meet the heterogeneous project requirements.

Spatial objects (such as points or cells) are used to simulate an individual's behaviour in geosimulation models, so that any patterns at a macro level can be revealed (Schoorman 2005). However, there are limitations for such spatial objects in simulation modelling, because they are not computing elements (Ahlqvist et al. 2005). The use of distributed GIS components will enhance the computing ability of geosimulation models. Also useful would be the ability to simulate dynamic processes and provide various services such as model management and data quality control (Batty et al. 2005). Moreover, distributed components will enable geosimulation models to be shared between users.

In principle, distributed component GIS is consistent with interoperability. With distributed components, it is less difficult to couple GIS with external models or tools, and they will thus work together in an interoperable framework. This provides a high degree of freedom to allow users to be able to couple specific components for their own purposes.

Traditionally, the strategy for tool coupling is to integrate distinct technologies within a common user interface; this can be compared with distributed component GIS which couples spatial tools in an open environment (Goodchild *et al.*, 1996). As the components are distributed, GIS techniques can be shared more widely through wireless mobile devices and the Internet. In addition, it can also be constructed as an open system on the Internet. Various tool developers would then be able to make their own contribution, resulting in there being diverse tools available for users to access and choose.

In conclusion, GIS architecture is evolving into a more open, interoperable, distributed and ubiquitous framework, being supported by various coupling strategies (such as the

Internet, Web 2.0, mobile devices, OpenGIS standards, distributed component computing and wireless communications). Distributed component GIS is the feasible result of these trends. It is considered that the more regular implementation of distributed component GIS within different applications will move these trends forward.

2.5.2 Developing Intelligent Capability for Distributed GIS Components

When provided with additional intelligence, distributed GIS components will become more powerful in computing, analysis and communication. Here, intelligent features could include proactive, reactive, adaptive and objective-driven abilities, as well as autonomous and communication abilities. GIS functionalities or techniques could then be decentralised into intelligent distributed GIS components. It would thus be more flexible and robust to use such decentralised and distributed components in the interoperable and open environments for different projects at different locations by different users. On the other hand, as more distributed components are developed, they will also be expected to self-control, as it is difficult to centrally control a large amount of distributed components in an open environment.

There are some technologies which have become well established in AI and have also been successfully applied in GIS. These technologies could be incorporated into distributed component GIS to develop the intelligence of GIS components. The first technology is 'fuzzy analysis', which shows strength when dealing with uncertainty (Pan *et al.*, 1998). To create an intelligent environment system, a Boolean value can never be reliable. By using the fuzzy sets to simulate the process of reasoning, improvements in decision-making can be achieved (Liao, 2004). Bringing it closer to the human capability of controversy, many options and decisions could be established by reasoning, giving the

best probable tactic to deal with a problem or rather what is most likely to be the correct answer to the question (Jamshidi *et al.*, 1997).

Fuzzy algorithms appear in many AI technologies, where they can particularly be applied to support agent learning ability. Nute *et al.* (2004) uses them to support goal analysis agents, to evaluate how well a management unit satisfied management goals by using four fuzzy categories (i.e. fails, nearly passes, barely passes and passes) that indicate the outcome of the agent action to the goal. From the outcome of current agent action, the authors set the algorithm to analyse a goal into desirable future conditions. One example is the agent-based intelligent infrastructure of the contingency management system, which is based on the cooperative game theoretical model with the fuzzy coalition's algorithm (Jacobi *et al.*, 2004; Sheremetov *et al.*, 2004). In GIS, fuzzy sets are used to handle spatial uncertainty. For example, Brimicombe (1998) introduced a method that was able to objectively handle linguistic hedges of uncertainty within GIS. This fuzzy set technique was propagated using Boolean operators, which can easily be translated back and forth into different, however real, languages.

The second technology is 'pattern recognition'. Journel (1981) introduced a form of the probabilistic classification approach, calling the indicator 'kriging', which Caers (2001) identifies as being two-point statistics that work on a pattern recognition mechanism. Caers (2001) used simulated images (which could be previous images of training data or new images of real or training data) and fitted in formulae to produce prediction values to a real image (data). This is achieved using different templates (scale of measurement but similar attributes) for a given location, starting at the central location [U](#) .

The third technology is the 'neural network'. A neural network or connectionism is the original AI technique proposed by Alan Turing (1940), who was the fore figure of computer

machines and AI. This technique works by imitating the human biological neural network (Bigus and Bigus, 2001; Li *et al.*, 2004). Using electronic signals similar to biological neurons, the intelligence is acquired by responding to these electrical signals and the system can manifest itself according to the condition perceived by the system (Turban and Aronson, 2001). Neural networks achieve their maximum performance through time. They normally require training to gain experience and to achieve their functional objectives. This training needs to be targeted at the goal in hand. With neural networks, Gilardi (2002) proposed a structure to handle geostatistical complexity using machine learning (ML) techniques. Caers (2001) also developed a structure to use neural networks for providing non-linear mapping data. Noh *et al.* (2000) recognises the limitations of using pure kriging techniques on complex data, and suggests the use of a neural network. He recommends the back propagation neural network for learning, as it can be controlled through the learning procedure; such that if provided with fewer learning cases, it can be kept generalised - as a result, this will produce far fewer hidden nodes and will minimise the 'grandmother' effect. He compares neural networks to 'decision trees'. Due to the adaptive ability of neural networks, he gives decision trees no chance in the competition for efficiency. However, data quality is often problematic when using a neural network. Another notable problem is the inability of neural networks to become accessible via interoperability.

The fourth technology is 'ontology', defined as:

"Theory that uses specific vocabularies to describe entities, classes, properties and functions related to certain views of the world which could be simple taxonomy, a lexicon or thesaurus, or even a fully axiomatised theory" (Fonseca *et al.*, 2002: p121).

Grubber (2002) further explains ontology as being the explicit specification of conceptualisation. In GIS, ontologies are seen by many researchers as the way forward

in dealing with interoperability (Smith and Mark, 1998; Fonseca and Egenhofar, 1999). To provide this, Nolan *et al.* (2001) suggest using agents as the desired technology to enable interoperability.

“The ontology is comprised of three critical components necessary to exchange information between GIS-domain agents, systems, or organizations, including: vector, raster, and image data; algorithms descriptions including name, inputs, outputs, and required parameters; and query/result information” (Nolan *et al.*, 2001: p99).

To achieve interoperability, they used ontology as the foundation for communication between agents. In AI, ontology is regarded as a specific reality that is described by an engineering artefact (Guarino, 1998), while others characterise it as a particular vocabulary that describes and reflects a specific view of the world (Fonseca *et al.*, 2002). In the context of this thesis, the definition for ontology will embody both the classical (which is also defined by GIS) and that of AI.

The fifth technology is Intelligent Agent technology, which is one of the most important developments in computer software since the introduction of object orientation. An agent is an autonomous, problem-solving, encapsulated entity operating within an open and dynamic environment (Wooldridge and Jennings, 1995; Wooldridge, 2000; Jennings, 2001). Communication and autonomy are central features of an agent, although they may also have many other intelligent features in different applied areas (such as mobility, ontology, cloning, adaptation, collaboration, reactivity, social ability, activity and pro-activity).

The technology associated with such agents can be viewed from diverse perspectives. Agents can be used as a resource for the development of complex systems, or simply regarded as a design metaphor. They can be algorithms, specific techniques or

infrastructures (Luck *et al.*, 2003). With such powerful characteristics, GIS techniques or functionalities may be decentralised as collaborating agents, whilst other agents could be developed to become distributed components across a network. Working as intelligent distributed GIS components, such agents are both robust and flexible - robust for different applications, systems or locations (as well as being widely shared and having high accessibility), and flexible with regard to collaboration, coupling, interoperability, extension or modification.

This thesis intends to develop the intelligent ability of distributed GIS components, mainly on the basis of agent technology. Agent-based GIS functionalities and techniques may be regarded as a key way forward in the development of GIS. Because spatial dependency is a key issue of GIS (see sections 1.2 and 2.2), agent-based variogram modelling will be investigated to see how key GIS functionalities may be developed as intelligent distributed components.

Spatial relationships and the inherent spatial dependency (spatial autocorrelation) of geographically distributed phenomena are a fundamental aspect of spatial data mining, analysis and simulations (Berry *et al.*, 2008; Getis, 2008; Haining, 2009). As a fundamental tool in GIScience and one which is often difficult to use, modelling the variogram using intelligent distributed agents would be a good way of investigating the concept of distributed component GIS. Agent-based variogram modelling could be derived from a series of agents which are autonomous, collaborative and even mobile. Interface agents set up procedures and parameters to manage all other agents. Raw data input is dealt with by the Data Input agent, while the Variogram agent carries out the modelling. Under the Variogram agent there could be different Semivariogram agents available (such as the Exponential agent, Spherical agent and Linear agent). With the estimation of a variogram model, the Kriging agent then makes interpolated fields, while under the Kriging agent

there could also be simple Kriging agents and Cokriging agents and so on. The Data Output agent and Visualisation agent would then be able to export the results, using different formats as desired.

2.5.3 Research Structure of Agent-based Distributed Component GIS

This research will be based upon finding the right structure and mechanism to achieve a pure agent-based distributed component approach to GIS. Initially, desk-work will be undertaken to review the existing theories of GIS research on agents, as well as the current technology being proposed, its interpretation and the application of agents for GIS. The current limitations will be established and potential improvements will be structured. At this point, the venture into current agent research will be explored, to ascertain the reasons for agent limitations when developing distributed component GIS. Experimentation to provide potential solutions will be undertaken in order to achieve the most appropriate structure.

Several possible structures will be established and experiments will be conducted to test which structure is best. This will be achieved using existing published (solved) case studies previously presented by Isaak and Srivastava (1989) and Cressie (1993). The agent system will be tested to find out how close it gets to the results found by the professional, and whether it is able to effectively learn and improve on these results over time. Thereafter, a more complex situation will be introduced.

The structure of this research is illustrated in Figure 2.2. This structure will allow GIS tools and functions to be developed further through the use of agent software development. This will be investigated through the development of an agent-based variogram. The developed agents will be tested using data from known problems, for which the solutions

are already known and published, to ascertain their ability to solve problems and compare results with the human expert solution (this is the known solution). Furthermore, the complexity of the simulated problems will then be increased, so that the effects on the data can be observed. The variogram agent performance can then also be observed as the amount of data and information increases. This should show the limitations of the developed tool. Within this research, the problem is whether the ability of the variogram agent meets the design requirements in terms of problem-solving and learning mechanisms.

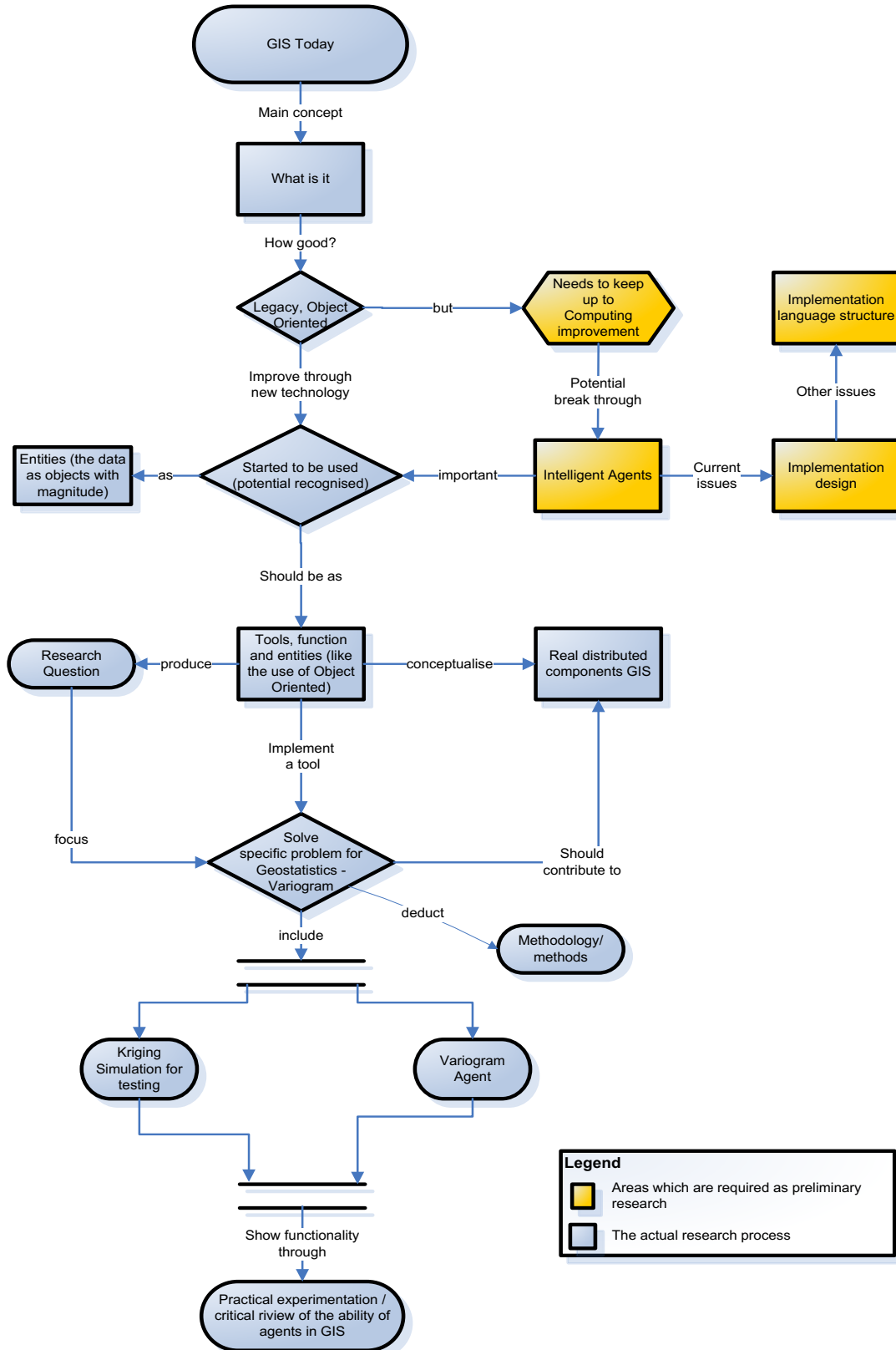


Figure 2.2: Research Structure for a Distributed Agent Platform

The specific outcome of this thesis is:

1. Agents are not necessarily object entities, but could be an environment (the tools that entities interact with and the interaction mechanism), and these would be beneficial in an environment notion to GIS.
2. Proof of concept that GIS tools can be developed using agent-oriented software development, just as they could be developed using object-oriented software development.
3. A structural architecture to developing agents in GIS components.
4. A fully implemented and workable variogram agent, as a proving ground for point 2.
5. A more robust late stage of the agent development cycle (refining implementation diagrams) to accommodate the ability of easily producing agents for complex systems.

The focus of this research is to create an architecture and investigation as to the importance of intelligent agents in GIS. This will be determined by producing a GIS component - the variogram modeller. The variogram was chosen to be the proof due to its modelling complexities and the current need for expert human interaction during the modelling.

CHAPTER THREE: AGENT-BASED TECHNOLOGIES

3.1 Introduction

This chapter looks into the origins of agents and, to some extent, their relationship to Artificial Intelligence (AI) and its origins. It will examine their characteristics and functionality, how agents were introduced and applied to different fields, and particularly agent technology applied to the field of GIS. The context of using agents in GIS and agents will be explored in relation to other AI and software development areas. Also in this Chapter different types of agents and their applications in the spatial domain are identified and described. Some challenges of deploying agents in GIS will also be presented.

3.2 Characteristics of Software Agents

Although agents are often defined and characterised in their own right as a discipline of software development, they were originally derived from AI, with arguably the first occurrence of the term 'agency' derived from John McCarthy in 1958. It all started when McCarthy was conducting trials to develop a system that aimed to function as a human. At that time, many of his peers did not even agree that computers could understand natural language, let alone function like humans. In 1952, Arthur Samuel had written the first algorithm to produce a machine learning program (the program was in the form of a game application). The program was received with huge acclaim and recognised as being responsible for an increase in the performance of checker players. In the 1960s, Bobrow (1964) from MIT, in his dissertation, proved that computers could solve the algebra of word problems well enough. This changed many researchers' conceptualisation of how computers could act as AI.

The agent concept in AI can also be seen in Hewitt's work (1977). Hewitt's concurrent actor model was based on Distributed Artificial Intelligence (DAI), with the idea to create entities that were self-contained within their internal state and which could respond to

messages from peer entities. These entities were also interactive and concurrently executing. From this point on, AI agents evolved and have formed into three broad areas: DAI, Distributed Problem Solving (DPS) and Parallel AI (PAI).

Substantial advances within the research field of agents started during the late 1980s, when the agent technology section of AI came into existence. Although non-human-like intelligent systems (often referred to as weak notions of agency, see below) still utilised agent technology; the main reason for the development of agents was to make AI effective and truly human-like. They were intended to solve problems associated with autonomy, ontology and mobility of the knowledge in AI systems. These characteristics, capabilities and functionalities of agents are what attracted research to use them in so many fields. Agents are characterised as being able to achieve communication with each other and with external users (like humans), and to harmonise a shared and common understanding of a domain (Lin *et al.*, 2001). With the increasing usage of distributed and ubiquitous computing, many researchers also point out the importance of interoperability (Wooldridge, 2003) and therefore the benefits of using software agents that can communicate, migrate and perform tasks across heterogeneous network systems.

Considerable research has been undertaken within software engineering, AI and other areas of software and hardware computing to develop suitable agents. However, as many researchers across the computing sector have shown interest in the subject, parallel studies have taken place and consequently many categories of results have been achieved.

Due to the applicability of agents in many different contexts and different circumstances, a number of definitions of 'agents' have been developed. Some of them describe agents as simple entities that can be developed simply by relying on the existing object and oriented by adding APIs (Okomoto 2009; Yu *et al.* 2009), while others describe agents in their own software engineering paradigms producing specific software designs that

should be able to conform to certain characteristics and often being more than just an entity (Jennings and Wooldridge, 2000; Shih, 2001; Wooldridge, 2003; Liao *et al.*, 2004). In addition, due to the word 'agent' not having a single standardised meaning and being in generic usage, researchers from diverse ranges of disciplines refer to agents differently according to their aims and purposes. For example, there are software agents that assist users when printing documents, and telephone and fax redialling agents (albeit that they have simple functionality).

"The metaphor has become so pervasive that we are waiting for some enterprising company to advertise its computer switches as empowerment agents" (Wayner and Joch, 1995: p95).

On the other hand, there is a range of agents with complex functions and intelligence characteristics, which can be mobile over networks. Thus, it is important to determine the underlying mechanism of agents as to be able to establish the degree of agency of an application.

Wooldridge and Jennings (1995, 2000) describe how agents are generally defined, particularly within AI research, as:

"...a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" (Wooldridge, 2002: p15).

Agents should be able to perform flexible autonomous actions to achieve a desired goal, with the basic properties of autonomy, interaction, reactivity and pro-activeness. This definition is perpetuated from similar definitions made by King (1995) and Nwana (1996). In general, a software agent can be defined and classified according to its characteristics and a notion referred to as 'agency' (how strongly the agent exhibits the agent characteristics). The following summarises the main characteristics of software agents:

- **Autonomy:** the capacity to act independently from external users and other agents, and in the sense of adding intelligence to users' instructions (Farjami *et al.*, 2000; Kwon and Lee, 2001; Callan, 2003; McGann *et al.*, 2009).

- **Mobility:** the ability to freely migrate from one host to another in a network and, when needed, initiate communication with other agents (Farjami *et al.*, 2000; Shih, 2001; van Breemen and de Vries, 2001; O'Grady and O'Hare, 2004; Amandi *et al.*, 2005).
- **Reactivity:** the capability by which agents can perceive their environment and respond to changes with actions in a timely fashion (Amandi *et al.*, 2005).
- **Social ability:** the ability to establish some type of agent-communication platform to interact with each other or, if required, with external objects (e.g. humans) (Hill *et al.*, 2004). This feature induces another feature, *collaborative behaviour*.
- **Collaborative behaviour:** the capability as such where each agent is given a discrete task, but must also work together to establish how they will share the information they collect for dealing with a collaborative job (Callan, 2003). This feature is often confused with social ability (described above); the difference is that collaborative behaviour is where an agent needs to communicate with its own kind, whilst social ability refers to the ability to propagate freely and pick up new knowledge on the way and even clone itself when required (Shehory and van Harmelen, 2004).
- **Adaptivity:** the ability to learn over time, as agents react to or interact with their external environment and change according to the experiences accumulated. Therefore, agents' performance should improve over time. This feature directly interacts with the reactivity feature. Agents that can adapt are sometimes called 'learning' agents (O'Grady and O'Hare, 2004). Adaptivity expresses the need for an agent to subsequently adapt to users (people) instead of the other way around (i.e. the agent should have a memory and so be able to learn and change according to experiences accumulated through time). Wooldridge and Jennings (1995) state that: "it would be impractical to assume that we could predict all possible events in the external environment and encode all the knowledge about those events in advance, agents need learning capabilities. How they react to new circumstances can be programmed, what they learn cannot".

- **Personalability:** the capacity to adapt to user needs. An agent should learn the pattern of user behaviour and mutate itself to do the specific function that a user would need (Shehory and van Harmelen, 2004). Being a software, but very deferent from normal software where information flow is only one way (from the user to the software), the agent should be fully interactive whereby the user can learn from it and *vice versa* (information flows both ways).
- **Pro-activity:** the ability to take initiatives without an external instruction from users, by using predictions. These predictions must reflect the true purpose of the agent's existence and should never work to manipulate the system otherwise (Amandi *et al.*, 2005).
- **Cloning ability:** the ability to replicate itself (Amandi *et al.*, 2005).
- **Ontology:** formally expressing the knowledge representations needed for a specific domain, which may in turn be seen as a single component (Gruber, 1993)
- **Intelligence:** generally refers to an agent's ability to learn and adapt to its environment.

Although those listed above are widely recognised characteristics of agents, there are still questions concerning whether some agents in particular disciplines and applications can be regarded as real agents. This confusion was pointed out by Sengupta and Sieber (2007) and is yet to be clarified. Agents can also be recognised as being

“a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future... [whilst a software program that] ...runs once and then goes into a coma, waiting to be called again... [can be told apart from an agent]” (Franklin and Greasser, 1996: p4).

Furthermore, many researchers have made an effort to classify agents, and to differentiate their software as one type of agent from other types. There have been many properties that have been used to identify different types of agents in the software industry. Thus, agents can often range from intelligent and collaborative ones to any

other autonomous software. Due to the wide range of agents, complications arise when differentiating types of agents. A generally accepted concept in grouping agents into those with a 'weak' or 'strong' notion of agency will be described below.

Agents can be separated into two groups: those with a weak notion of agency and those with a strong notion of agency; identified depending on the characteristics that such agents exhibit (Wooldridge and Jennings, 1995; Nwana, 1996). Agents with a weak notion of agency are defined according to their basic characteristics. There are four main characteristics:

- autonomy;
- reactivity;
- goal-oriented (pro-activeness); and
- social ability (being able to interact and communicate with other agents).

Software exhibiting these characteristics form the basis of being software agents. Entities that do not exhibit all of these four main characteristics are often questioned as to whether they can even fall into the non-agent category (Franklin and Graesser, 1996). However, the basic characteristics can also be regarded as to having a weak or strong notion, such as there are agents with a weak notion of autonomy and agents with a strong notion of autonomy (Sengupta and Sieber, 2007). Agents with a strong notion of agency are determined as possessing most of the agent characteristics previously described, in addition to the four main characteristics. In AI research areas, agents with a strong notion of agency are those regarded as holding certain properties (either conceptualised or implemented) which are generally applied to humans. These agents can have notions of belief, intention, knowledge and obligation, or even emotion (Shoham, 1993). Such strong notion agents can exhibit characteristics such as mobility, capability of collaborative behaviour and the ability to adapt, learn and be rational. Based on these two groups of agents, a third type of agent can be identified, called specialised agents (Al-Zakwani *et al.*, 2007a). These specialised agents can exhibit either a weak

or strong notion of agency (from the above definition), but possess only the specific features that are required for these agents to perform their delegated tasks.

A distinction between different types of agents can be made according to the characteristics that the agents exhibit (Nwana, 1996). Nwana (2006) has divided the past agent research into two strands: deliberative and collaborative agents. Since the late 1970s, there was an emphasis mainly on deliberative types of agents and macro issues such as interaction and communication between agents, distribution of tasks, coordination and negotiation resolution. The deliberative type of agent has been defined as those agents that have an explicitly represented and symbolic model of the world and make their decisions through symbolic reasoning (Wooldridge & Jennings, 1995). The second strand, which started during the 1990s, covers a much broader and diversified range of agents, including collaborative, interface, mobile, information, reactive, hybrid and smart agents. In this broader range of agents, different types of agents are distinguished depending on the particular characteristic(s) on which the agents have more emphasis. For example, collaborative agents have more emphasis on the characteristics of autonomy and cooperation, while having less emphasis on other characteristics. In the same way, interface agents focus more on autonomy and learning than collaborative agents. Truly smart agents should have the characteristics of autonomy, cooperation and learning equally, but as Nwana (2006) points out, this is more of an aspiration than reality.

Moreover, because of the wide range of agent applications, the types of agents are also often differentiated according to the function and tasks carried out by them. One example is the role-specific classification of agents provided by King (1995) includes navigation, help, management, search and retrieval, role-playing, domain-specific, analysis and design, development and testing agents - and many others. Another example is the agent typology discussed by Dillenbourg (1999) includes personal agents, assisting decision-making agents, agents assisting human-computer interaction (HCI) and agents assisting spatial data retrieval. Agents in different applications can

also range from those with a weak notion of agency to those with a strong notion of agency.

As discussed above, software agents can be classified by the characteristics which they possess, varying in agency from being weak to strong. This also depends on which function the agents are being employed to perform in their environment.

3.3 Some Applications of Software Agents

Since the 1990s, applications of agent technologies have been found within a wide range of disciplines, such as computer networking, software engineering, AI, human-computer interaction, mobile systems, control systems, decision support and electronic commerce. In addition, agent technologies have also been utilised in the disciplines of economics, social science, philosophy and logics. Such examples show the diversity of agent technology, its research and applications. An agent is regarded as a design metaphor or a source of technology for development applications (Luck *et al.*, 2003). An agent, as a design metaphor, can offer a way of developing applications around autonomous communicative elements and complex systems constructed with software tools and infrastructure (Jennings, 2001). In application development, agent technologies can be a source of technologies and algorithms for dealing with interactions in dynamic and open environments.

Applications of agent technologies can be categorised differently, due to the wide range of application fields and the different ways by which they can be grouped. Some applications are grouped according to the main roles undertaken, such as the four application areas highlighted by Haag and Cummings (2007).

- buyer agents that emphasise those applications using agents to retrieve information around a network related to products and services, and to monitor buying preferences and buying histories, then offering customised services;

- user agents which stress those applications and use intelligent agents to take actions on a user's behalf;
- predictive agents refer to monitoring and surveillance applications, often undertaking tasks such as observing and reporting on software and physical equipment functionality;
- data mining agents that highlight those applications utilising agents to operate in a data warehouse for knowledge and information discovery, in order to detect major shifts in trends, a key indicator, or the presence of new information.

The applications of agent technologies can also be broadly categorised as assistant agents, multi-agent decision systems and multi-agent simulation systems (Luck *et al.*, 2003). The first category emphasises assistant agents, which mainly uses agents to gather information and execute transactions on the Internet on behalf of users. The second application area focuses on multi-agent decision systems, which utilise agents in the system to cooperate with each other and make joint decisions. One example could be the joint decision-making mechanism used in auction applications (such as those used in eBay), where there are mainly two kinds of agents being used, the negotiation agent and the auctioneer agent. The third application category mainly concerns multi-agent simulation systems, which use agents as models for simulating real-world domains. These multi-agent models usually deploy various components, and interact in complex and diverse ways. Examples of such applications can be found in a wide range of areas, such as human economics and social science, biological populations, road traffic systems, computer networks and computer games.

As we can see, this way of categorising agent applications depends on how agents are perceived to behave. Between the first category and other two, the distinction is that one is a single agent and the other two are multi-agent systems. Even though agents in the first category of applications may well need to interact with other agents, multi-agent applications place more emphasis on collectively taken decisions rather than individual ones. The second category of applications (multi-agent decision systems) is aimed at

systems that comprise of agents and what agents do, whilst the third category is aimed at the understanding derived from the system and an appropriate representation of real-world components that could be provided by agents. Therefore, the main difference between these two multi-agent system applications is that one is used for taking decisions while another is for understanding the phenomena.

In practice agents have been studied extensively for supply chain management by Lin *et. al* (2001), Jankowska *et al.* (2007), Nienaber and Barnard (2007), Chan and Grosz (2008) and IBM (2008) and many others. All these authors agree that the supply chain management will benefit by using a combination of proactive and reactive strategies that keep critical supply processes available without interruption. Nienaber and Barnard (2007) used software agents to support processes for project management. To study the effect of software agents to influence market behaviour of human traders, Grossklags and Schmidt (2006) have experimented in a laboratory, a MAS-based platform for a double auction market where passive agents with an arbitrage-seeking strategy are used with human subjects. They found that common knowledge about the presence of software agents triggers more efficient market prices. Even though, they explained that "controlling for information on software agents' participation, the introduction of software agents results in lower market efficiency". Zivan *et. al* (2009) have developed a team agent to represent mobile sensors who can adjust their locations with respect to changes on their current perception. The system presents a new information exploration method using Distributed Constraint Optimization Problems (DCOP) algorithm where the credibility of agents themselves is confirmed using reputation model (Huynh *et. al* 2006). The authors explain the model as capable of handling "a dynamic problem in which the alternative assignments for agents and set of neighbours, derive from their physical location which is 'also' dynamic." This has achieved the goal of evaluation and adjustment of sensors during the deployment phase to correspond to the dynamic changes of the environment acted upon. An example of this is the peer-to-peer environment (Yu *et. al* 2009). Yu *et. al* (2009) have introduced agents capable of learning to discover service providers on peer-to-peer network. They

act as processes for message propagation on a distributed message relaying to remove the problems on communications or/and long response time. Similar applications have been developed to help simulate and improve team performance on time stressed group tasks by ad-hoc decentralisation of human teams (Sukthankar *et. al* 2009). Sukthankar *et. al* (2009) explain that the simulation studies how human teams rise to the challenge by analysing “the communication patterns of teams performing a collaborative search task that recreates some of the cognitive difficulties faced by teams during search and rescue operations”.

Okamoto *et. al* (2009) have developed personal assistants using agent. The system was able to determine the requirements for hierarchical organizations and horizontal organizations by comparing the agent performance on assisting the impact of the personal assistant have on the behaviour. They have found that, for hierarchical organisations, the agents can assist on improving load balancing through task allocation and failure recovery while for horizontal organizations by improve communication. They concur that experiments showed system facilitate most beneficial to horizontal organizations. The algorithm the agents used in Okamoto *et. al* (2009) are similar to that of Lieberman *et. al* (2001, 2006) who came up with a concept for integrated annotation and retrieval of images using agents. Based on the user's everyday work, a proactive user-interface agent automatically searches an image library and seeks chances for image annotation and retrieval. The main function of the agent is to facilitate the finding and usage of the images. The agent monitors the typing of user's text editors. Thus images relevant to the current text can be inserted in a single operation. Lieberman *et. al* (2006) assert this by saying "descriptions of images for storytelling can be seamlessly employed as raw material for annotation. Common-sense knowledge about situations in which pictures are taken, described, or used can help provide semi-automatic annotation and indirect inference for retrieval." Furthermore, Nealon and Moreno (2003) have used Agent-Based Applications in health care. They build: an Agent-Based Community Care Demonstrator using a Worldwide Agent Platform; Agent-Based User Interface Adaptivity in a Medical Decision Support System; a Multi-Agent

System for Organ Transplant Management. Schweiger *et. al* (2007) have emphasized this importance of agents in healthcare due to their characteristic of proactivity and flexibility which the authors have affirm to be the dominant characteristics in healthcare.

One widely used area of agent application involves those used in e-commerce. One such example is in Trading Agent Competition (TAC), in which agents are utilised to find and book hotels and make travel arrangements (Greenwald and Stone, 2001). Another example is deploying agents in a network (e.g. the Internet), to retrieve relevant information for their users about products and services, such as certain types of personal shopping assistant which are able to search online stores for product availability and price information, electronic marketplaces in which agents buy/sell goods and so on. One of the best known examples of these kinds of agents is when buyers/sellers are in the presence of Amazon.com, where agents are used to monitor user buying preferences, the types of items or themes which have been purchased in the past, so that a list of customised items that the user might like to buy can be offered.

Agents can also be utilised for those applications where everyday mundane functions are performed automatically on an individual's behalf. For instance, agents can be used for managing one's e-mails, such as checking for e-mails and re-arranging the mail according to the owner's usual preference (e.g. all mail directly from work should come highest on the priority reading list, while suspect e-mails should be put on the lowest priority list), answering standard e-mails (e.g. greeting e-mails and reminders from other colleagues) and making sure that in turn it alerts the owner of any answered e-mails. These agents could also schedule meetings, by negotiating a suitable time, and one of this kind of agents is offered as a standard example by the JADE engine (JADE Manual, 2004).

Agents are often used in applications which assist the filtering of information. One such example is through the deployment of agents to assemble customised news reports used by the CNN website. CNN has a system where users get assigned to an agent

which will arrange the news according to user registered profiles and preferences. Also with CNN, there are agents that can discuss certain topics with you (such as sports and betting) and agents that can find information for you on the subject of your choice, scanning the web pages to look for and highlight text that constitutes the 'important' part of that information.

Agents have been widely applied in the area of manufacturing, including the configuration and collaborative design of products, scheduling and controlling manufacturing operations and production sequences, controlling robots, and process control by autonomous reactive systems. One example is a software platform known as ARCHON, which is used to build multi-agent systems and has been applied in electricity transportation management and particle accelerator control. Such agent systems have also been applied to monitor and diagnose faults in nuclear power plants, spacecraft control, climate control and steel coil processing control. Another application which uses agent technologies is for prediction and monitoring, such as those agents used in NASA's Jet Propulsion Laboratory for monitoring the inventory, planning, scheduling equipment orders and food storage facilities. These agents can monitor complex computer networks. Developed in the BRAHMS programming environment (Sierhuis, 2007), they allow computer networks to keep track of the configuration of each machine connected to it.

Multi-agent systems have also been deployed for applications for allocating resources and managing the operation across a telecommunications network, where agents represent a range of components of the network. Agent technologies have been employed with considerable effort in the telecommunications industry, particularly since 1992. For example, the programs ACTS and EURESCOM in Europe (by BT, Telecom Italia, Telefónica, Portugal Telecom and Telia) have carried out specific research on agent applications to telecommunications services, service management and workflow, and methodologies for agent development. Another example is in forecasting the traffic on telecommunication networks, using a multi-agent system simulating user behaviour.

Agents have often been used in entertainment, such as developing computer games. Agents can be used to assist playing computer games, and computer games agents (especially intelligent and mobile agents) are becoming one of the main trends in this area (Al-Zakwani, 2006, 2007b). An example of such computer games is SimCity which is described by Maxis (2002). A rich simulated environment can also be developed with a range of synthetic agents, where a user can interact in real-time. One example of using agents to create remarkable scenes is in the movie 'The Lord of the Rings'. The agents used can learn over time and their behaviour can change, thus the movement and action of each individual can perceive and respond to the environment and other agents, and convincing effects were achieved by the autonomous actions of agents. Agents are also used to facilitate intelligence and reactions in some specialised environments in computer games. For example, the First Person Shooter (FPS) genre gives computer-controlled characters realistic behaviour (e.g. by dodging bullets, or working together in a team to kill the human players) (Mateas, 2003). The Belief, Desire, Intention (BDI) agent architecture (Bresciani, 2004) can be used to accomplish this. A Belief is the knowledge that an agent has about itself and its environment (O'Hare and Jennings, 1996), such as how to stay alive and how to kill an opponent. Desire is the outcome or goal that the agent would like to bring about (Shajari and Ghorbani, 2004b).

Simulation applications of multi-agent systems, in general, aim to represent real-world environments with an appropriate degree of complexity and dynamism; such applications can be the simulation of economies, societies and biological environments. For example, multi-agent simulation has been used for analysing climate change, capturing the development of social pressures such as the outcome of individual choices and social interaction, or researching the impact of change on various biological populations. In social simulations, two types of approaches are often taken. One type aims to establish logical systems to underlie social interaction, whilst the other observes and models social processes. These two types can work together.

Thus, it can be seen that there has been a wide range of agent applications introduced over recent years, of which this section has only shown a few examples. Furthermore, in the computing industry, agent-based software development and tool coupling is becoming a preferred consideration. What is not discussed in this section are the applications of agent technologies in GIScience (i.e. in spatial phenomena), which will be the focus of the following sections. Although there have been ranges of agent applications in GIScience, geosimulation has been one of the main areas in agent technology application up to now.

3.4 Use of Agents in Geosimulation

3.4.1 Early Use of Cellular Automata in Geosimulation

The concept of geosimulation originated from urban studies. In order to deal with the complexity of urban phenomena, individually-based modelling technologies were introduced (for example, cellular automata [CA]). Urban simulation models were developed, based on individual urban objects such as households and pedestrians. To build such models, there was the need to develop tools with which to construct a spatial structure and consider the spatial behaviour of individual objects. Torrens and O'Sullivan (2001) devised the term 'geosimulation' to cover this new field of modelling. The geosimulation concept is widely applicable from urban study to ecology, economics, social science and other disciplines where spatial individual-based models could be applied (Benenson and Torrens, 2004a).

One influential technology in geosimulation, as mentioned above, is CA. The roots of CA can be traced back to Basic Automata (BA) in AI. BA is a simple processing mechanism that operates on a rule-based structure. It is composed of states, an input stream, rules and a timer (Turing, 1940; Von Neumann and Burks, 1966). In a rule-based structure, an automaton receives an input and acts to that input by using a set of rules. An example of this is 'if today is Sunday, then switch the heater off'. This is achieved by a mathematical construct that can be explained using set implication rules.

In principle, CA works just like BA with spatial implication. A collection of CA forms a lattice structure, where each cellular automaton influences its neighbouring cellular automaton (see Figure 3.1 for a simple visualisation).

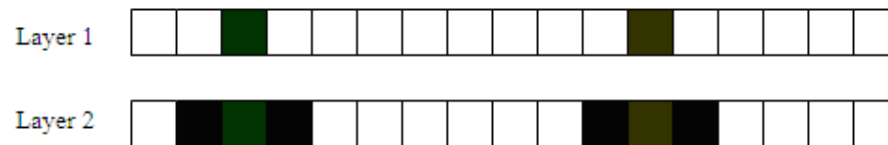


Figure 3.1: Demonstration of Cellular Automata

Figure 3.1 shows a structure having a cell being operated (covered in black) and shows the influence of and on the nearby cells. This could be regarded as a simple two dimensional (2D) CA - as such, on the first run (layer 1), we have picked the cells we want to work on. The next run (layer 2) captures the two closest cells (which are to the left and right of the original cells). How the cell selection works is governed by sets of rules, which are similar to the Turing machine rules of automaton structure (Turing, 1940). Each cell has a state, condition and action, and thus synchronously these cells could be transient such that the cell state is altered according to the action of a cell close by. The action is governed by a set of rules similar to those provided by the game platform.

In John Conway's *Game of Life* by Gardner (1970), cells have an 'alive' or 'dead' state where, by using the rules he provides, for each iteration the cells' states change accordingly. These rules are stated by Gardner (1970) as:

- Survivals - every counter with two or three neighbouring counters survives for the next generation.
- Deaths - each counter with four or more neighbours dies (is removed) from overpopulation. Every counter with one neighbour or none dies from isolation.

- Births - Each empty cell adjacent to exactly three neighbours (no more, no fewer) is a birth cell. A counter is placed on it at the next move.

These rules in CA have also been discussed in the geosimulation area, such as

“a 'live' cell in one iteration remains 'alive' in the next iteration if its neighbourhood contains either two or three other alive cells, otherwise its state is coded as 'dead'. Additionally, a 'dead' cell in one iteration can become 'alive' in the next if its neighbourhood contains exactly three 'live' cells” (Torrens, 2003: p 10).

There is more in-depth discussion on CA and its uses in Batty's “Cities and Complexity” (2005). Given the important position of CA in geosimulation, the above discussion reveals the natural links between concepts/technologies in geosimulation and AI.

Geosimulation focuses on space and geography, which distinguishes geosimulation from other simulation methodologies. It has been well applied to simulating an individual object's behaviour in order to study the spatial and dynamic phenomena at above-individual. As discussed earlier in Chapter 2, the development of geosimulation is technically heterogeneous, which can be a collection of geocomputational and visualisation tools, spatial data representations/models, interoperable systems or practical approaches (Albrecht, 2005). CA-based modelling initially inspired the concept of geosimulation. Agent-based modelling provides the opportunity to enhance and broaden geosimulation.

3.4.2 Next Stage of Using Agents in Geosimulation

Applications of geosimulation can be found in diverse areas, such as urban planning, urban crowds, environment and ecology, hazards and contingency, retailing and business geography, as well as wayfinding, security and evacuation. The following are some examples to demonstrate the evolving use of agents in geosimulation.

One area, which geosimulation is being applied, is in modelling sprawling urban growth. For example, Benenson and Torrens (2004) developed SprawlSim, using the structure of CA with some agent interaction mechanisms. This structure allows the simulation to be able to change its environment (using automaton) and adapt to the environment (using agents).

Moulin *et al.* (2003) used a multi-agent system-based geosimulation to improve crowd simulations. The simulation is based on agents interacting in virtual geographic environments based on 2D and 3D platform. The agents have spatial cognitive capabilities which include perception, navigation and reasoning. They are able to perceive the terrain characteristics (elevation and slopes), the landscape surrounding the agent (including buildings and static objects) and other nearby agents. The environment is given real life environments that includes a smoky area un/pleasant odors, distractive events occurring in the agent's vicinity like explosion, etc. The agents are able to utilise these perceptions to their advantage. Moulin *et al.* (2004) also developed PADI-Simul, which is an agent-based geosimulation software supporting the design of geographic spaces. Bédard *et al.* (2003) identified AI to improve decision-support tools that would facilitate geographic knowledge discovery. In practice, this improved On-Line Analytical Processing (OLAP) into Spatial OLAP. The system was intended to improve and facilitate access and exploration of environmental and health geospatial data to aid health specialists. Bédard *et al.* (2003) explains the main function of the project as a "help on reducing health risks caused by an environmental source by providing a quick and easy access to high quality environmental and health data to improve decision-making and interventions, access to statistics and other information and the discovery of new knowledge". The system was AI based but not exhibiting much of software agency, however agency was fully incorporated in Chaker *et al.* (2009) to increase the system robustness and to allow simulation of environments which allowed to capture health issue and contaminations on real life environment over simulated human displacement. SimWalk (2007) is another geosimulation-based experiment to enhance the security of public transport (train and bus) travellers. SimWalk is an open-

source tool used to model pedestrian flow and define security strongholds for certain activities. It is also used to model urban planning and evacuation. It is based on a generic platform where any pedestrian situation can have its problems modelled and simulated. One of the cases that was modelled using the SimWalk is the Islamic pilgrimage in Mecca. The simulation studied and found solutions to the crowd flow control related problems. The persistent problems are based on pedestrian flow for the circumambulation of the Ka'aba. The entry into the circumambulation area is channelled by multiple entry points and thus causes large number of pilgrims to flow in from multiple directions at the same time. This causes congestion and end up with disasters as pedestrians push into each other. SimWalk has managed to introduce a solution for this crowd control by employing agent with reactivity character and allow them to flow into the platform modelled Ka'aba. The right model was determined by introducing and studying the influence of obstacles helped on crowd control for the pilgrimage through experimenting on area extension on Levels of Service at the circumambulation area.

NED-2 is a simulation system which aims to be intelligent and goal specific (Twery *et al.*, 2004). It integrates tools that are designed to monitor the ecosystem of a forest and make intelligent decisions when needed by integrating with tools like GIS models of wildlife and vegetation growth. NED-2 is knowledge-based, which uses semi-autonomous agents to manage these tools for the user by building prescriptive management plans (Rauscher *et al.*, 2000). It was originally designed to provide silvicultural prescriptions to meet specific goals on timber production (e.g., to diagnose forest health problems) (Nute *et al.*, 2004).

Using geosimulation, Gosselin *et al.* (2005) have examined the expansion of the West Nile Virus (WNV), which is at its worst level of incidence since 1999. This research commenced when the public health authorities in North America built and operated a surveillance system to try to contain the spread of the infection so as to reach their territories through bird (carriers of the virus) migration. Bouden *et al.* (2005) suggest using geosimulation to simulate the behaviour of mosquitoes and corvidae which are

related to the spread and transmission of the WNV. They realised the need for reliable forecast of the closest possible level of expected risk and the time of occurrence to establish preventive measures to tackle the epidemic. This requirement for a more robust interpretation and prediction served to be the reason to refer to the Multi-Agent GeoSimulation. This simulation takes place in a virtual mapping environment representing a physical geographical environment with climate being factored in. Given that birds are the carriers of the virus to the Northern America they were developed as the vector of the Agent Based system that simulates their interactions to the mosquitoes over space and time in relation to the spread and transmission of virus.

More advanced 2D and 3D multi-agent geosimulation arose in 2005. For example, knowledge-based agents are used in modelling customers' shopping behaviour in a shopping mall (Ali *et al.*, 2005). The simulation was initially based on a 2D graphical platform where individual agents have knowledge based on the shopping behaviour of a real population of shoppers. The agents were equipped with a capability to act upon their surrounding and making decisions in a 'micro-scale geographic environment', a shopping mall. The individual agent behaviour is based upon a real agent (a person) who have had their behaviour captured through an interview and embedded into the software agent. Thus based on the given characteristic, agents were able to choose the right shops to go to and so demonstrate decision making and navigation character of real human. Given the real user knowledge of the shops arrangement in the mall, the agent was able to start shopping based on its position in the mall exhibiting perception character, its knowledge of locations in the mall using memorisation characteristic. The agents can realise a closer shop while heading to a different shop and change their course of action which again show decision making, perception and memory capabilities. The agent will go around the mall until it has completed its shopping task or run out of time. The simulation was then turned into 3D to allow a better capturing of this realism behaviour by allowing the agent to have real life manoeuvres (real life environment in 3D).

To support human planning and spatial cognition, Sahli (2005) has also experimented on agent-based geosimulation. Due to the limitation of real life evacuation planning and in general on tackling wild fires, a close to real life software system is needed for this purpose. Given the uncertainty of wild fire, Sahli (2005) has implemented a Multi Agent Geo-simulation with agents capable of planning through anticipating a change of scenario and using reaction characteristic of agency. The simulation was also designed on 3D environment to support real life perceptions (provide all geographical angles a human can perceive).

3.4.3 Issues in the Use of Agents for Geosimulation

There has been an enormous increase in data availability over the past decade in terms of data type, volume, coverage and scale. Such a trend undoubtedly enhances the capability of geosimulation and has led to many new models being developed.

With large amounts of data and diverse models, spatial phenomena can now be studied from various perspectives and in considerable depth. Meanwhile, large amounts of data and different models might increase the demand for data exploration, data and model management. This poses a challenge in managing data and models in geosimulation such as handling large sets of spatial data both off-line and on-line, also managing a range of software and tools.

Distributed components across the Internet offer both opportunities and challenges for geosimulation research. Multi-agent systems have been used to query and integrate distributed environmental information over a network (Purvis *et al.*, 2003). Such a system is a collection of collaborating agents. A query from the user agent is passed to the query agents. The query agents send the query to data source agents that contain information on sources of data. Sengupta and Bennett (2003) have designed an agent-based framework to utilise online data and models for spatial decision support. Their agents assist users to locate and retrieve spatial data and analytical models distributed

on the Internet. The agents then automatically transform spatial data ready for inputting into analytical models through the use of GIS software. NED-2 (Nute *et al.*, 2004) mentioned in the last section, is another case, in which a set of semi-autonomous agents can set up and run external simulation models, load rule-based models, set up and execute external GIS, visualise outcomes and generate hypertext reports. However, deploying distributed components over a network to achieve geosimulation is still an area which needs much more research.

Geosimulation is recognised as a new field of simulation modelling, which covers various geographic data and spatial analytical techniques. A range of data quality problems then arise which can have a serious detrimental effect on the fitness-for-use of model outputs. Errors and uncertainties would be introduced in data collection, transformation, integration and manipulation. Errors and uncertainties might also propagate through different phases of modelling. Simulation outcomes could be misrepresented in visualisation and mislead the decision-making. Therefore, the awareness of spatial data quality is vital for the further development of geosimulation. Effective solutions are needed for managing the spatial data quality in geosimulation modelling. Research has been carried out using agent technology to control spatial data quality in geosimulation modelling by Li (2006, 2009).

Validation is another important issue for every type of modelling. For geosimulation modelling, there are some technical barriers for validation as well as verification, calibration or evaluation in general. This is because currently most geosimulation models are individual-based models which normally have a large number of dynamic spatial objects. To control the behaviour of these objects, various parameters and random factors are involved. There is also often a lack of empirical data as reference. Under such circumstances, traditional validation methods are often computationally heavy, time consuming and might sometimes even be infeasible. Li *et al.* (2008) have developed an innovative agent-based service solution for the validation and calibration in geosimulation modelling. On the other hand, if there are empirical data available,

geosimulation provides an opportunity to validate new theories or concepts. More research is required in this area of study.

At present, there are not many open-source geosimulation models available for use. However, as a new field of simulation modelling, open-source models will help to disseminate novel achievements in research and promote their applications to industry by sharing new ideas and techniques. Particularly, one strong point of geosimulation is bringing various technologies and tools together. Open-source models will certainly encourage contributions from different areas and disciplines.

3.5 Classification of Agent Deployment in the Spatial Domain

3.5.1 Understanding of Agency

The notion of applying software agents in GIS is by no means a new idea, as can be seen earlier in this chapter regarding geosimulation. They have also been extensively recommended as non-simulation entities to aid analysis. The use of agents has been discussed in the context of GIScience by many researchers such as Openshaw and Openshaw (1997), Goodchild (2004), Torrens (2004b), Reitsma and Albrecht (2004), Albrecht (2005) and Batty (2005a). Agents have been studied in many applications to facilitate GIS. The main focus can be deduced from the agent definition itself, of being a piece of software that can travel across networks autonomously and perform its required function by reacting to a particular condition. Agent technology has enabled greater interoperability and has offered the ability to work with large distributed processes and data. However, it is still important to understand agency in the context of the applications of agent technologies in spatial domain. Such understanding can largely be based on the type of agent being deployed, which can be defined by the characteristics of the agent at work.

In much GIScience research, it is not uncommon to see the idea of agents being bound together with objects. In GIScience (and particularly research into geosimulation), there seems to be a direct perception of an agent in relation to an object, such that they certainly coexist in every situation. This can be seen in the early agent-based applications in GIScience. As Benenson and Torrens (2004b) state, “geosimulation models are noteworthy in their depiction of simulated entities”. However, in computing, there is a clear difference between object and agent, although they can be coupled together. In one example, the use of object-oriented programming (OOP) can be seen in the CLIMEX tool for modelling global climate change, which is implemented using two types of objects: the spatial object and thematic object (Fedra, 1996). In another example, Faulkner (1999) has taken the research approach of writing Java classes to deal with the variogram. At this point we can see that the object-oriented functionality of GIS and strong computing mechanism of fully utilising agents has not yet been met. To move forward, firstly it would be helpful to clarify the differences and draw a clear boundary between object-oriented and agent-oriented software development. The issue of object- and agent-oriented technologies also exists in the studies on distributed component GIS. It is therefore necessary to have a look into different types of agency, as used by researchers in their applications.

Starting with the agents that have a very weak notion of agency, like those mentioned by Franklin and Graesser (1996), there are agents that can have some degree of automation but do not qualify for one fully functional characteristic of an agent. Such agents might possess hypothetical features that make them look like an agent (an example of this is the paper clip that appears on MS Office, where if a user makes multiple mistakes or appears to be stuck, the clip will appear to offer help). Conversely, there are agents with a strong notion of agency (even sometimes referred to as futuristic agents), like those discussed by King (1995), which provide for a role-specific classification of agents. The types of agents defined include navigation, help, management, search and retrieval, role-playing, domain-specific, development, analysis, design and testing agents..

Many researchers in GIScience area have successfully implemented agents as objects in simulations, such as human individuals, land parcels, fluid flow and landslides. However, in the pure computing environment and especially software development, agents have been extensively used within systems that perform some functional work. Those agents often poses learning and social collaboration features. Take the example of a timetabling agent, it can make sure that there is no clash within the timetable, by understanding and applying the user's preferences. Another example of agent application in software development can be seen in Braciani *et al.* (2002), where an ice producing company had their system implemented using an agent-based software development paradigm. The system was intended to provide an easy process for the ordering, manufacturing and delivering of ice-based goods. It is important to bring this notion of agency into to GIS applications.

3.5.2 Classifying Agent Usage in the Spatial Domain

As already discussed, agents have been deployed in the spatial domain. Such agents are termed here as geospatial agents. Sengupta and Sieber (2007) argue that there are some geospatial agents which fit within the AI paradigm of agents. Based on the classification scheme proposed by Franklin and Graesser (1996), they refer to the two types of agents: artificial life geospatial agents (ALGA); software geospatial agents (SGA). ALGA simulate the behavioural response of an individual to an external stimulus using available computational models of rational decision-making behaviour. ALGA are grouped into five themes: agricultural particles/subsidy, human movement, human networks, animal movements, land use and land cover. SGA are designed to act autonomously on behalf of an entity to manage geographically explicit information. An entity can be a person, other SGA, or a piece of software or hardware. In other words, SGA assist people in managing information and making decisions in hardware and software environments.

A new approach is proposed in this thesis to classify the use of geospatial agents. It considers three dimensions: the notion of agency, the model of spatial data, how agents are applied in GIS, particularly in areas of geocomputation and geosimulation. This classification aims to be comprehensive and practical, and more importantly it aims to improve the understanding of agent-based applications in spatial domains. It will also be helpful in dealing with issues of agent-based applications in some future developments, such as spatial data quality, model validation, integration or collaboration of different types of software and technology. In the proposed classification, the agents utilised in spatial domains are grouped by specification, as follows:

- Role of the agent: agents acting as object-entity, processes or service;
- GIS data structure: agents acting on tessellation or vector;
- Properties of agents: agents having basic features or extra features.

Where agents are deployed as object-entities, these can be points, lines or polygons which may represent individual persons, road segments or land parcels. When it acts as a process, it will concern itself with the dynamic state changes or physical movement over time. The simulation of geographic processes can often be seen in hydrological, landslide or pollution modelling. One example is oil spill modelling (Li, 2001, 2006). In oil spill modelling, decision-makers are not only interested in where the oil spill starts and might end, they also need to know how the spilled oil is transported, as well as the change of thickness, the process of evaporation and emulsion of the spilled oil. When agents act as services, they can control the activities provided to support mapping, modelling or analysis, which could include data searching and retrieval, data quality testing and assessment, model calibration or management of a model system. Services can be tools, solutions or platforms. With GIS data structures broadly using tessellation or vector, applications using agents therefore operate either on tessellations (as in CA) or on vectors (as in network applications).

The basic features of agents are those previously defined as having a weak notion of agency which normally includes autonomy, communication, reactivity and pro-activity.

Extra features to impart a strong notion of agency would include additional functionality such as intelligence and mobility. These extra features can make a key difference in agent usage towards real distributed GIS and can result in a major shift towards functional spatial analysis with a reduced input from human expert.

Figure 3.2 shows the proposed classification along three axes:

- Role of the agent (agents acting as object-entity, processes or service) in which 'O' represents object-entity, 'P' represents process, 'S' represents service;
- GIS data structure (agents acting on tessellation or vector) in which 'T' represents tessellation, 'V' represents vector;
- Properties of agents (agents having basic features or extra features) in which 'B' represents basic features of agents and 'E' represents extra features of agents.

Table 3.1 lists some examples of utilising agents in the spatial domain, which are classified according to this classification scheme.

The actual classification is derived from an agent that can exhibit more than one of these classes and the degree of agency is identified through the degree of its exhibiting the features that establish individual classification while the inclusion of extra features define.

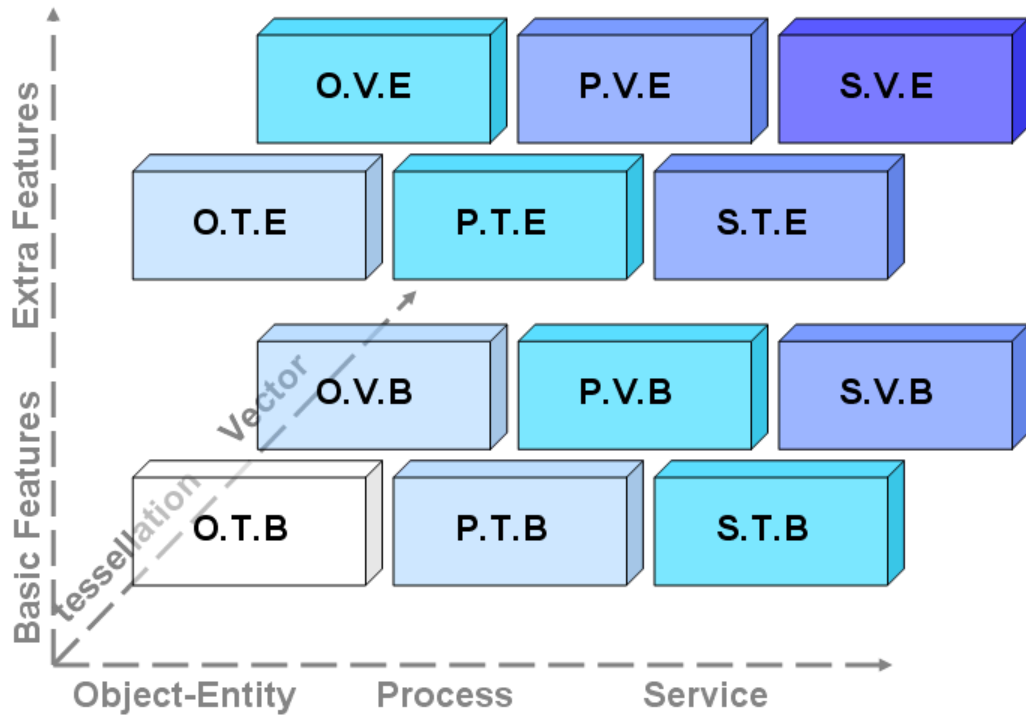


Figure 3.2: Proposed Classification of Utilising Agents in the Spatial Domain

Table 3.1: Examples of Utilising Agents in the Spatial Domain

Author	Agent Type	Discussion
Heppenstall et al. (2005)	O.T.B . O.V.B .	A Multi-Agent system on spatial interaction model where the agents are actual objects (petrol pumps with the main variable being the price (for petrol price setting)) to be studied. The data type is classified in both analog vector and tessellation creating 2 parallel agent models. The agents used in this system exhibit basic features where reactivity and proactiveness play the main role of agency.
Benenson & Torrens (2004b)	O.T.E .	The system uses automaton as object and introduces the intelligence agency character to simulate environment (urban phenomenon) using cellular automata concept of AI

		and a tessellated environment as the canvas for the automata growth and multiplication.
Torren (2006 and 2007a)	O.T.E . O.V.E .	Torrens (2007a) explains the system as “a reusable platform for modelling human behaviour, action, and interaction in social and anti-social crowds, for the purposes of simulating a variety of behavioural, human, and urban geography scenarios”. The system has two types of interaction as far as agent technology is concerned. Those that act from point to point and refer to as vector and those that act on within their boundary of movement. The example of vector action of the agents is demonstrated when an individual agent being the object having specific tasks to perform in time while for the tessellation is when the crowd acting upon their given confinement of space. The agent in this simulation exhibit extra agency features using Cellular Automata for intelligence and has the capability of navigation. Apart from modelling crowd behaviour the author has used the system for residential relocation behaviour and megacity modelling.
Batty (2005a)	O.T.E .	The concept presented by Batty (2005a) is focused towards developing a model for simulating the evolution of cities. The agent usage is presented on objects acting upon their confined spaces, like the tiles in tessellation model. Batty (2005a) explains this as “units of space ‘which’ are conceived as cells and populations as individual agents”. The agents are presented as pedestrian movement on a tessellation of building scale. The system is then able to present a view of a city at a regional scale with the effect of individuals (real human represented as agent in the system) responding to environmental attributes encoded in cellular (tessellation) landscapes. The system also is used to model urban growth at the city scale.
Brown et al. (2005)	O.T.B ., O.V.B ., P.T.B.	The system defined by Brown et al. (2005) uses spatial data as objects (agents) and spatial process (agents) for exploration and explanation of spatial-temporal phenomena. Thus presenting agent in two forms; object and process on the same platform. The agent representing

	, P.V.B.	process on space representing tessellation while agent representing object act as vector. The agents act on a tessellation for that they are capable of acting on the given agenda on a given field (tile). This system has managed to showcase the a model for aiding tight coupling for geographic data using agent-based process which uses four key geographic data features: identity, causal, temporal and topological.
Reitsma and Albrecht (2005)	P.T.B. , P.V.B.	The paper describes the development of a novel methodological approach for simulating geographic processes. A system is developed that represents data model as processes (as agents). This methodology complements existing approaches to dynamic modelling, which focus on the states of the system at each time step, by storing and representing the processes that are implicit in within a given model. The processes act upon tessellation grids and on objects representing vectors. The system uses data model (called nen) which Reitsma and Albrecht (2007) constructed them to focus on “existing modelling approaches on representing and storing process information, which provides advantages for querying and analyzing processes”. The nen is a basic agent with pre-programmed functionality and does not have any of the advance characteristic.
Tail et al. (2004)	P.V.B.	The research covers the use of Borehole Optimisation System (BOS) as a tool for informing decision-makers and demonstrate its applicability for the development of urban groundwater resources. The BOS itself is an API of ArcView where it integrate three GIS areas: Catchment Zone Probability Model (CZPM); the Land-use Model (LM); and the Pollution Risk Model (PRM). The system identifies current and historical industries located within the selected probabilistic catchment zone and simulate a process (agents) by using industrial and the associated hydrogeological and contaminant data (as objects) and predict probabilistic contaminant concentrations.

Li, Brimicombe & Li (2008)	S.V.B.	The authors cover the argument of actual implementation of the agents based GIS to help solve problems related to complexity of models that have large numbers of parameters, for validation and calibration. They suggest using Multi-Agent System which can run large number of models as a solution where agents act as service providers for assistance. The agents achieve this by using their collaborative characteristic to perform calibration and sensitivity analysis for the model validation. The agents are basic entities (objects) that are given parameters for navigation on a vector pattern. The focus here is the concept of the Multi-Agent platform providing service of validation to other GIS Agent based systems. The system explained here is to provide a proof of concept for collaboration of agents to provide closest possible pattern to reality for model building using MAS in GIS.
Al-Zakwani et al. (2007a)	S.T.E. , S.V.E.	The authors explain the concept of agency for assisting in complexity reduction when dealing with spatial data (object and especially process). The Multi Agent System employs the intelligence and reactivity characteristic of agency to help analysing geostatistical data. The ability of an agent to mimic human analysis and prediction is experimented using Geostatistics tool, Varigram. Thus the MAS (in this case known as Variogram Agent) is itself within agent environment which offer service to the expert Geostatistics user. The agent are demonstrated to work on a tessellation or/and vector model as long a human user can perform similar task. Thus the agent is to act on the place of the human user during data analysis (even though not entirely replacing them).

3.6 **Demands and Gaps in Deploying Agent Technology in GIS**

As discussed in previous sections of this chapter, software agents have established abilities in computing. Originally developed for AI, they can be applied in various areas

of different disciplines. Software agent technologies have been applied in GIS to simulate the object-entities and dynamic geographic process. Agents have demonstrated their strength of autonomy, social ability (i.e. communication) and other characteristics in spatial data modelling. In the field of geosimulation, there are many good examples of agent-based urban, environmental and socio-economic modelling. Meanwhile, as we know from previous discussions, there are various types of agents. Some of them have a weak notion of agency, whilst those may have a strong notion of agency.

With the expansion of geosimulation, many issues arise relating to data and modelling such as data exploration and analysis, data management and sharing, data quality, model management and sharing, model validation and calibration. Therefore, there are increasing demands for advanced technologies to deal with these data and modelling issues and, furthermore, to develop stronger geosimulation models or spatial techniques in the broad sense.

Currently, various techniques have been introduced into the research area of GIScience from the area of AI. For example, Brimicombe (1998) developed a method with fuzzy sets, which is able to objectively handle linguistic hedges of uncertainty within GIS. This fuzzy set technique is propagated using Boolean operators which can be easily translated back and forth into different languages. Using neural networks and machine learning (ML) algorithms, Gilardi (2002) proposed a structure to handle the complexity of geostatistics using ML techniques. These techniques could be very helpful when working with homogenous data and given resources, but they lack the strength to expand and share resources. Although they are currently being used in GIS, these techniques are mostly applied in the internal mechanisms of GIS tools rather than devised as generic tools. For example, ArcGIS has some notion of neural networks, but does not possess any functions (i.e. does not have a mechanism specifically for neural networks).

Agents have the potential ability to deal with the abovementioned issues in geosimulation. For instance, they are able to manage sharing data, tools and models over a network or by wireless. They can also track down errors and uncertainties in spatial data modelling and rectify rules, parameters and data, which will save huge amounts of labour and time. Such agents are those with a strong notion of agency, but which have not been widely used in geosimulation research and applications. A large number of agent applications in current geosimulation research are mainly reactive, with basic agent features although there is the potential to incorporate a great degree of agency in agent-based modelling. Agents with a strong notion of agency are strong in terms of autonomy, reactivity, pro-activity and social ability (communication). They normally also have other characteristics; for example, they could learn and adapt to keep up with the required capacity of decision-making mechanisms in the ever-changing GIS environment, whilst travelling over networks (mobility) and thinking independently. As they are able to overcome the difficult data and modelling issues in geosimulation and consequently are able to effectively decentralise and distribute computing tasks, agents with a strong notion of agency can be developed to support distributed component GIS. However, much work still needs to be undertaken in this direction.

In the proposed classification of agents in the spatial domain (see Figure 3.2), we can see that the majority of agent-based models in geosimulation belong to the classes of agents acting as object entities. There are fewer agent-based models where agents acting as services. Agents acting as a service should have a strong notion of agency. This is because these agents need to take on heavy and complex tasks, such as data quality testing, model calibration or wayfinding computation. There is currently an apparent lack of attention being paid to developing such service agents for GIS and geosimulation. This thesis aims to fill that gap by creating and investigating an intelligent, service-oriented varigram agent.

GIS software and related analytic tools are usually complex and require large amount of computing power. They often require skilled human in order to run the more advanced

functionality. In this thesis, we have proposed using software agents to experimentally facilitate modelling variogram, which is often considered to be one of the more difficult items to achieve without using a human expert. The major problems are the size of the tools, the size of the data, the chaining of the process and the constant requirement for input from expert users. Agents with reactivity, mobility, social ability and other intelligent features would seem to be good candidates to deal with these issues.

CHAPTER FOUR: METHODS FOR MODELLING THE VARIOGRAM

4.1 Introduction

This chapter looks into the concept of geostatistics and its principal techniques. The main focus in this Chapter is on modelling the variogram. The use of variogram is an important technique for spatial prediction and thus the emphasis here will be to understand the process and procedure with related mathematical formula for the modelling. Presented here are the steps normally taken by an analyst which in this research are intended to be learned by an agent. Thus the process will be discussed from the data cleaning and sampling to fitting the right model of the variogram and subsequent kriging. The main purpose here is to examine the features required for an agent to be able to start functioning as a variogram modeller, acting in a similar way to a human analyst.

4.2 Geostatistics

The concept of spatial analysis refers to a set of formal techniques which study georeferenced data where geostatistics is a key set of techniques (Grumpecht, 2009). Since the work of Krige (1951) and further popularisation by Matheron (1962) for geostatistical techniques, a revolution had occurred which has produced contributions from a number of disciplines focusing on spatial phenomena. For support of this point Getis (2008) observed that "by the 1990s, the field of spatial analysis had matured to the point where the methods and concepts it created were becoming fundamental to researchers in a host of disciplines including geography, ecology, epidemiology, sociology, urban planning, geology, and environmental studies". Geostatistics can be defined as models and methods to analyse and study the regionalised structure and

degree of spatial dependency in geographical data (Goodchild, 1993; Lee *et al.*, 2001). Deutsch (2002) states that geostatistics is “a study of phenomena that vary in space and/or time”. Isaaks and Srivastava (1989) regard geostatistics as offering “a way of describing the spatial continuity of natural phenomena and provides adaptations of classical regression techniques to take advantage of this continuity.”

Geostatistics can also be viewed as a collection of numerical and mathematical techniques dealing with the characterization of spatial phenomena as a sub-set of spatial statistics. An important aspect of spatial statistics is the study of spatial autocorrelation (Bailey and Gatrell 1998, Ripley 1981), similar to correlation statistics. The difference of correlation statistics to spatial autocorrelation is that the former is designed to show relationships between or/and among variables while the latter shows the correlation within variables across geographical space (Cliff and Ord 1981; Hubert *et al.*, 1981; Getis 2008). Geostatistics are fundamentally embedded into the concept of spatial autocorrelation (Cliff and Ord 1981; Cressie 1993). Given the ever growing usage of georeferenced data, Getis states the importance of spatial autocorrelation as playing "a crucial role among spatial modelers" and as a

"fundamental element of all spatial models" central to model building. He explains this importance as it "provides tests on model misspecification; determines the strength of the spatial effects on any variable in the model; allows for tests on assumptions of spatial stationarity and spatial heterogeneity; finds the possible dependent relationship that a realization of a variable may have on other realizations; identifies the role that distance decay or spatial interaction might have on any spatial autoregressive model; helps to recognize the influence that the geometry of spatial units under study might have on the realizations of a variable; allows us to identify the strength of associations among realizations of a variable between spatial units; gives us the means to test hypotheses about spatial relationships; gives us the opportunity to weigh the importance of temporal effects; provides a focus on a spatial unit to better understand the effect that it might have on other units and vice versa 'local spatial autocorrelation'; helps in the study of outliers". (2008: p291)

Geostatistics has the ability of offering a description of spatial continuity and the spatial dependency including how the formulae can be determined and be simplified to provide a base model for agent-based distributed component GIS. Spatial dependency is always present in geographical phenomena as a prerequisite of landscapes having form and process; otherwise the data under study would be random on the plane (\mathfrak{R}) implying an absence of process (Atkinson and Tate 2000). In order to describe the spatial relations and correlations of observed variables, there are three principal techniques used in geostatistics (Burrough and McDonnell 1998):

1. Correlogram: graph of the autocorrelations (ρ_l) against lags (l), where autocorrelation refers to the correlation of a point and its neighbour(s) either in time and/or space (Cressie 1993) and where lag refers to increasing separation between groups of points in time and/or space (Atkinson and Tate 2000). The correlogram is a measure of autocorrelation which is also referred to as a “serial correlation” (Cliff and Ord, 1981).
2. Semivariogram: A technique used to establish spatial dependence between values at spatial locations. This measure is commonly referred to as the variogram, and will be so called throughout this thesis. The variogram is discussed in detail in Section 4.3.
3. Covariance: A statistical measure of the correlation between two variables. In geostatistics, the covariance is usually treated as the inverse of the variogram, computed as the overall sample variance minus the semivariance (γ) at each lag.

Although a number of mathematical tools that could be used for data descriptions will be discussed in this chapter, modelling the variogram will be the principal focus of interest in

this research. The modelling of the variogram is the key function in geostatistics (Burrough and McDonnell, 1998). It is used to detect and describe spatial dependencies and their patterns in space (Ripley 1981; Isaak and Srivastava 1989; Dungan et al. 2002). This is achieved by inspecting the variogram either through the shape of the plotted semivariences or from the type of theoretical model that could best fit the plotted values (Rossi et al. 1992). The main theoretical models and their interpretations will be discussed in detail on section 4.3. The variogram model can be used to interpolate the surveyed data points and thereby model the spatial phenomenon as a continuous surface or field. The process of interpolation is commonly referred to as Kriging.

The technique of kriging (Krige, 1951; further discussed in Section 4.6), which uses parameters of the variogram, can be used to interpolate the data or simply to provide confidence in the description of the analysed data. The latter is done by calculating residual errors after interpolation – usually to a held back sample. Kriging is defined as a technique of statistical modelling that can interpolate unknown data values of a continuous surface given a set of known data points. In general geostatistical estimation is done in a two-stage process:

1. Collected data are studied to analyse the predictability of the missing values. This is normally done through the variogram. At this stage the variogram is used to model the difference between values of one location to another location given the distance and direction.
2. Missing values are predicted at un-sampled locations along with a measure of variance at each location. This can be done using different forms of Kriging (Cressie 1993, Isaak and Srivastava 1989). The most commonly used is Ordinary Kriging.

This two-stage process can be carried out using a number of available Geostatistical tools. Given the focus of variogram modelling these tools are included in GSLib (and its descendants), Avenue scripts in ArcView, VarioWin, Java Class and the ArcGIS Geostatistics which has been moved to agent technology since ESRI's adoption of RePast (Najlis and North, 2005). GSLib (Deutsch and Journel 1992) started as a Microsoft DOS command line for analysing data and later moved to Graphical User Interface (GUI) which provides an user-friendly interface. GSLib was originally developed using structural form of programming (Fortran) with a heavy code for execution (i.e. requires a lot of computing resources) and minimal reuse capability. Currently it is based on semi-object oriented programming (OOP) which has increased its reuse capabilities. However, even with this new development, the tool is still too large and heavy to run and acts monolithically which, introduces limitations on the amount of data that can be processed at one time.

VarioWin is another tool developed using structured programming but, unlike GSLib, it is constrained to building the variogram. With this tool an expert user needs to do all the required exploratory data analysis (e.g. remove outliers, check for distribution type and organise the data) before creating the variogram (experimental or model). However VarioWin provides an easy mechanism for constructing the variogram by providing a function to dynamically change the lag. The main problem with VarioWin is that it is limited to small datasets and the human expert is considered to be the main driver of data analysis prior to variogram modelling.

More sophisticated is the Java Class for Variogram by Faulkner (1999) where the entire program was developed using OOP which provides easy extension and reusability. The tool focuses on producing variograms from pre-analysed data, which makes it similar to the previously discussed VarioWin.

A sophisticated tool in Software Engineering terms is ArcGIS Geostatistics with RePast as its companion (Tatara *et al.*, 2007). RePast is an agent plug-in that can assist data analysis and improve functionality for geostatistics (North and Macal, 2007). However, the tool requires an expert to determine the parameters and define the queries for the execution.

The above tools are different as each was developed using its own programming language and its functionality. However their functionality has similar characteristics in that they are monolithic and single function tools.

Another tool that uses agent technology for spatial data analysis (though not geostatistics) is Oracle Spatial which provides software agents for analytical and statistical tasks in a similar environment to that of RePast. These two agent tools differ on their complexity in execution and running speed which will be examined in more detail in Chapter 6.

As discussed in Chapters 2 and 3, due to the introduction of the Internet and technologies like Web 2.0, the computer environment and information management has moved from being monolithic to distributed. There is also an increase in on-line data availability which has had a huge impact on the uptake of GIS and the use of on-line GI Services (e.g. Google Maps). In most cases when dealing with data describing spatial phenomena, geostatistical tools provide analyses of the structure of the data (regionalised variables). With the implementations of geostatistical tools described above, the user currently needs to have available (and needs to know how to use) the individual tools as well as knowing how to structure the data and format them for passing from one tool to another. This requires expert knowledge of all the tools.

This is where tools which provide functionality and flexibility as a distributed environment

will have greater advantage and thus be more desirable. The suggestion here is to use software agents to achieve this mechanism. Software agents can offer services to one another (agent to agent), to external tools (to aid interoperability e.g. Oracle Spatial to VarioWin) and to users (easing their analytical tasks). The introduction of this kind of technology will see a major shift into more effective use of distributed spatial data and running GIS from thin clients and mobile devices.

This is the aim of implementing a variogram agent so as to investigate the feasibility and implications of distributed component GIS. Compared to traditional monolithic GIS tools it should provide faster data analysis (time taken for all steps of the analysis); better functionality (more functions within a perceived single tool, since there will be many tools coming together seamlessly to create an environment that appears to be a single tool); less complexity (most of the inputs required will be learnt by the agents and expert user interaction will be minimised); and finally more consistent results as data will be handled in a single format throughout the analysis (see Chapter 6).

4.3 The Variogram

Modelling the variogram is a fundamental technique in geostatistics, which is used to measure the spatial structure of observations (attributes of point measurements). Knowledge of the spatial structure as modelled by the variogram also allows prediction of unsampled points within the reference frame (\mathfrak{R}). This is because each attribute observation (z) is tied to its x,y coordinate location, expressed as $z(x)$. The calculation of the variance at each spatial lag (h) is divided by two to give the semivariance (which is the main point of interest) and the plot of semivariance against the lag forms the semivariogram. However, in common usage within the literature, the term 'semivariogram' is shortened to 'variogram'. The semivariance is expressed as:

$$\gamma(h) = \frac{1}{2N(h)} \sum_{i=1}^n (z(x_i) - z(x_j))^2 \quad (4.1)$$

...where h is spatial lag, N(h) is number of data points within lag h, z is the variable being modelled as determined at points x_i and x_j (Barnes, 2002).

The semivariance quantifies spatial correlation and is plotted for all h to form the variogram. A variogram can be defined as a tool that is capable of characterising the spatial data. A variogram can be described as:

“...a statistically-based, quantitative, description of a surface's roughness. Which is a function of a separation vector: this includes both distance and direction, or a Dx and a Dy. The variogram function yields the average dissimilarity between points separated by the specified vector (dissimilarity is measured by the squared difference in the Z-values)”
Barnes (2002: p. 16).

However it could be thought of as a statistical function borrowing expressions from probability to find the degree of correlation between two or more points residing on a spatial plane (\Re).

A variogram is a measure of the variance between data as a function of distance. This function is crucial in geostatistics and as a predictor of data values in general (Cressie, 1993). It is one of the means of determining the spatial correlation of a regionalised variable. A variogram has three important parameters:

1. **The nugget:** this is the point where there is no dependence on the point being paired, or there is spatially unexplainable dependant variation (perhaps due to measurement error), or if purely random variance is being experienced in a model. This is being defined mathematically as the positive intercept of the variogram to

the ordinate (Cressie, 1993). Atkinson and Tate (2000) define the nugget as the point where the semivariance is constant throughout the lags. This is expressed as c_0 .

2. **The sill:** this is the value at which the dependency is none existent or at a minimum and thus the variogram levels out (Isaaks and Srivastava, 1991). The sill can be defined numerically as:

$$sill = c_0 + c_1 \quad (4.2)$$

....where c_1 is the structured variance and represents the spatial variation that is spatially dependent (autocorrelated) at a relevant scale as defined by the lags (Atkinson and Tate, 2000).

3. **The range:** this is simply defined as the distance or lag which marks the end of the obvious dependency. In general it is the inflection where the model reaches or approaches the sill (particularly in reference to spherical, exponential and Gaussian models). The nugget model has a constant sill and a range of zero and the linear and power models use the sill and range to simply to define the shape of the slope (see section 4.3.1.).

4.3.1 Modelling the Variogram

One of the challenges of the variogram is associated with choosing the appropriate model for the data at hand. The experimental variogram is a line graph linking the semivariences (γ) for each successive lag. On visual examination of experimental variogram, an appropriate model can be fitted which mathematically describes the structure of the data.

Therefore, a variogram model is a simplification of the spatial structure of the data. Parameters of the variogram can be used to interpolate values at unsampled points. Within modelling, spatial structure can bring insight into what we want to study and understand the relationships between the attributes of neighbouring and nearby points in the study of local processes. Identifying the right model is referred to as 'fitting the variogram'. There are two branches of such models: unbounded and bounded (discussed separately below).

4.3.1.1 Unbounded

These are variograms plotted using models that have an infinite semivariance as the lag (h) increases (Atkinson and Tate, 2000). These models are generally referred to as power models. There are three basic types of unbounded models: linear, logarithmic and power.

- **Linear model:** exhibits the same rate of increase between semivariance to lag (h), thus there is no sill. An example of this fitted model is given in Figure 4.1(a)
- **Logarithmic model:** the semivariance rate of increase is slightly higher than the lag (h) which is referred to as logarithmic model. The graph of this model is shown in Figure 4.1(b)
- **Power model:** the semivariance rate of increase is slightly lower than the lag (h) which is referred to as power model. Figure 4.1(c) shows examples of curves for powers >1, powers <1; for power = 1 the fit is a straight line and the fitted model becomes linear.

Mathematically these three models show the correlation increasing indefinitely. These models are defined by the following formula:

$$g(h) = c \cdot h^\omega \quad \text{where } 0 < \omega < 2 \quad (4.3)$$

....where h represents lag distance, a represents range, and c represents sill

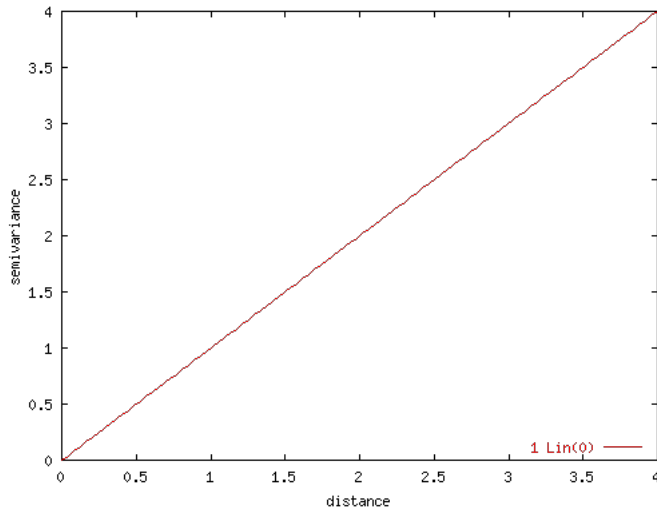


Figure 4.1(a): Variogram unbounded model:

Linear

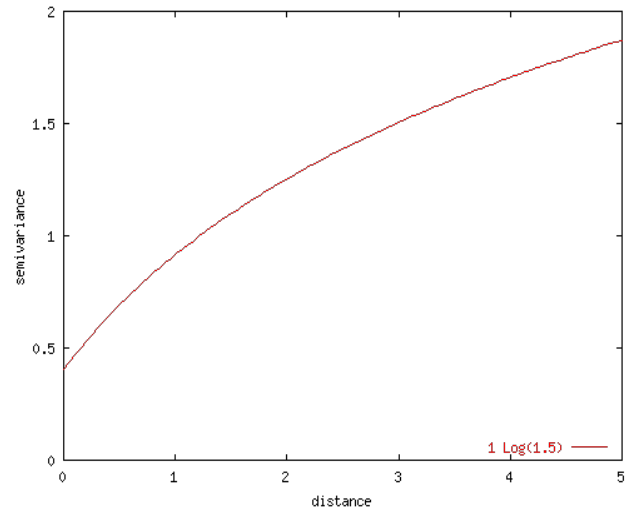


Figure 4.1(b): Variogram unbounded model:

Power

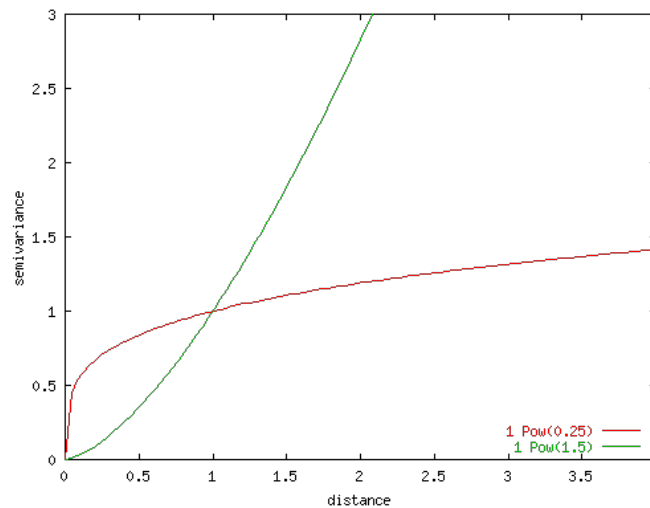


Figure 4.1(c): Variogram unbounded model:

Logarithmic

Source: Pebesma (1999).

Figure 4.1: Variogram, Unbounded Model

4.3.1.2 Bounded

These are models that have an identifiable sill (Cressie 1993). The parameters that identify the important variogram components are h , a and c where h represents lag distance, a represents range, and c represents sill. These models are:

- **Exponential Model:** a function frequently used when fitting mathematical models to an experimental variogram, often in combination with a nugget model. This model is defined by formula below and presented in Figure 4.2(a)

$$g(h) = c. \left(1 - \exp\left(\frac{-kh}{a}\right)\right) \quad (4.4)$$

...where k is a constant; k=1 or k=3. This is often used for those models having a large nugget value and a relatively long range.

- **Gaussian Model:** a function frequently used when fitting mathematical models to an experimental variogram, often in combination with a nugget model. This model is defined by following formula and presented in Figure 4.2(b)

$$g(h) = c. \left(1 - \exp\left(\frac{-kh^2}{a^2}\right)\right) \quad (4.5)$$

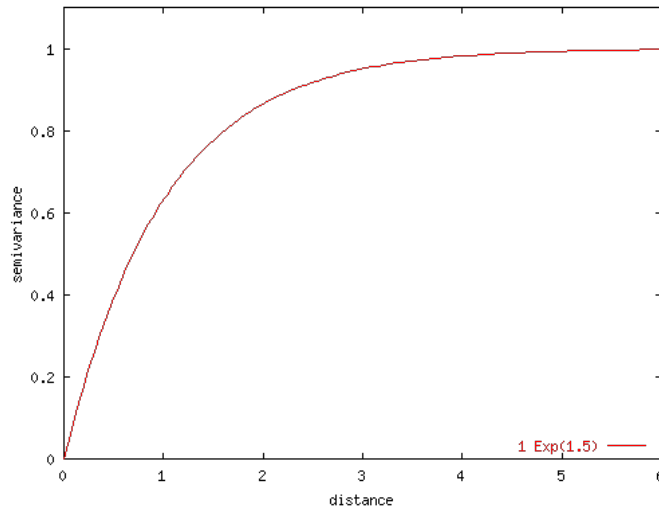
...where k is a constant; often k=3. This model gives a more 'S' shaped curve and is often not stable for models without a nugget.

- **Pure Nugget Model:** a constant variance model shown in Figure 4.2(c), which can be used in combination with one or more other models. This model is defined as:

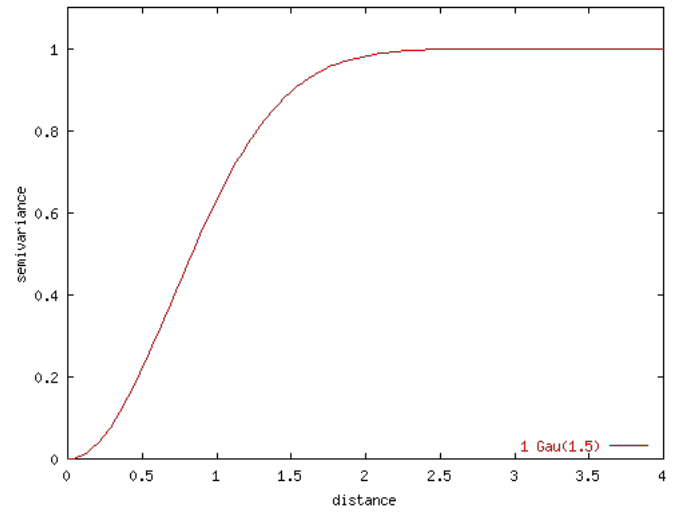
$$g(h) = \begin{cases} 0 & \text{if } h = 0 \\ c & \text{otherwise} \end{cases} \quad (4.6)$$

- **Spherical Model:** a function that is frequently used when the nugget effect is important. This model is defined by formula below and shown in Figure 4.2(d)

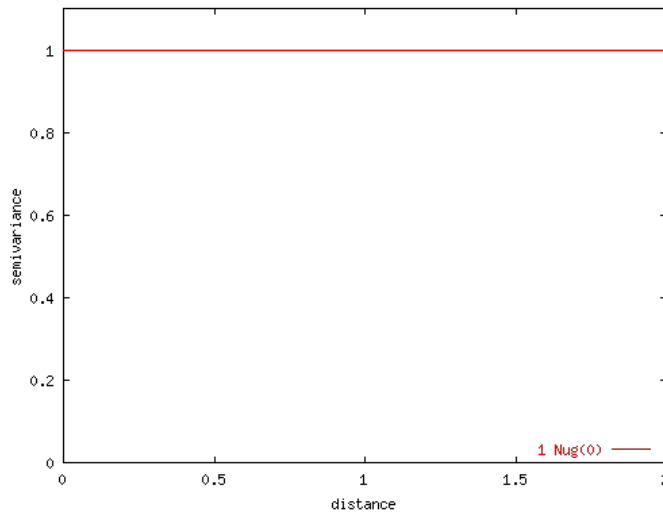
$$g(h) = \begin{cases} c \cdot \left(1.5 \left(\frac{h}{a}\right) - 0.5 \left(\frac{h}{a}\right)^3\right) & \text{if } h \leq a \\ c & \text{otherwise} \end{cases} \quad (4.7)$$



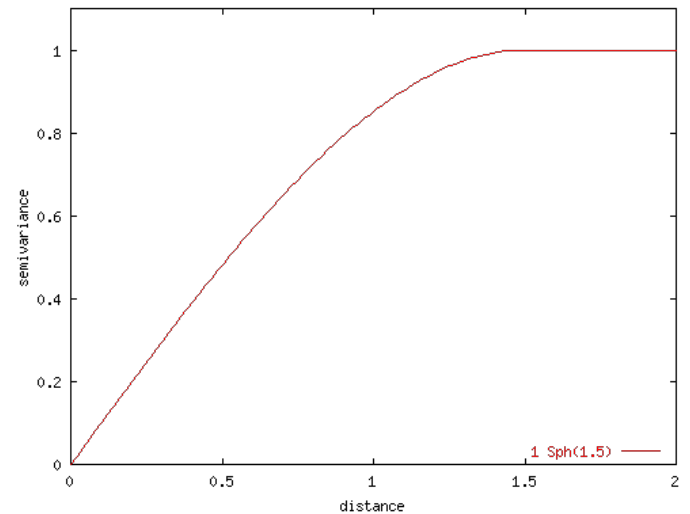
**Figure 4.2(a) Variogram bounded model:
Exponential**



**Figure 4.2(b) Variogram bounded model:
Gaussian**



**Figure 4.2(c) Variogram bounded model:
Pure Nugget**



**Figure 4.2(d) Variogram bounded model:
Spherical**

Source: Pebesma (1999).

Figure 4.2: Variogram, Bounded Model

4.4 Data Aspects of Modelling the Variogram

The main function of the variogram is to provide a description of the spatial structure of geographically distributed data. Prior to plotting the experimental variogram and fitting a model, there are a number of data preparation steps that are necessary depending on the nature of the data.. These are data sampling, data polishing including outlier removal, identifying and removing patterns such as trends and clusters. The order in which they are carried out depends on the particular data set to be analysed.

4.4.1 Data sampling

Sample data are a subset of an exhaustive data set where the exhaustive data are the total collected data for the area under study. Data sampling is usually deployed as a means of reducing the exhaustive data set without altering its integrity and information content and yet reducing the computational overhead (Isaaks and Srivastava, 1989). This may be necessary for example when modelling from remote sensing data. Ripley (1981) defined the standard sampling schemes as:

1. Systematic:

- a. Centric, in which a grid of equally sized squares known as quadrats is placed over the area to be sampled where elements closest to the centre of each square is selected.
- b. Non-aligned, in which a grid of equally sized squares known as quadrats is placed where equal number of element are randomly selected in each cell.

2. Random:

- a. Uniform, in which each element of the data has an equal chance of being selected.

- b. Stratified, in which first the data is characterised then divided into homogeneous group according to certain characteristic then a sampling is performed in within these groups.

Kent and Coker (1998) suggest that stratification should be used when there are major and/or very obvious variations within the surface under study. The sampling strategies can vary depending on the objectives of study and the nature of the surface being modelled. To limit the complication on investigating an agent-based approach in this research, only mathematically defined strategies are used. These are quasi-random (selecting specific point/cell locations within an existing data set), stratified-random (proportionate sampling from a number of classes) and cluster deduction sampling (for removing spatial bias from an existing data set; see Section 4.4.4).

4.4.2 Polishing and Outlier Removal

It is not unusual that data are collected by means which cannot guarantee their consistency within a specified range of accuracy. Thus the data need to be polished (cleaned) to increase their reliability and fitness for purpose. There are several predefined techniques that include:

- Outlier analysis:
 - boxplot
 - Grubbs test
 - scatter plot (used for bivariate description)
- Trend analysis:
 - trend regression techniques

- weighted median polish
- Cluster analysis:
 - determine the presence of cluster
 - k means cluster analysis.

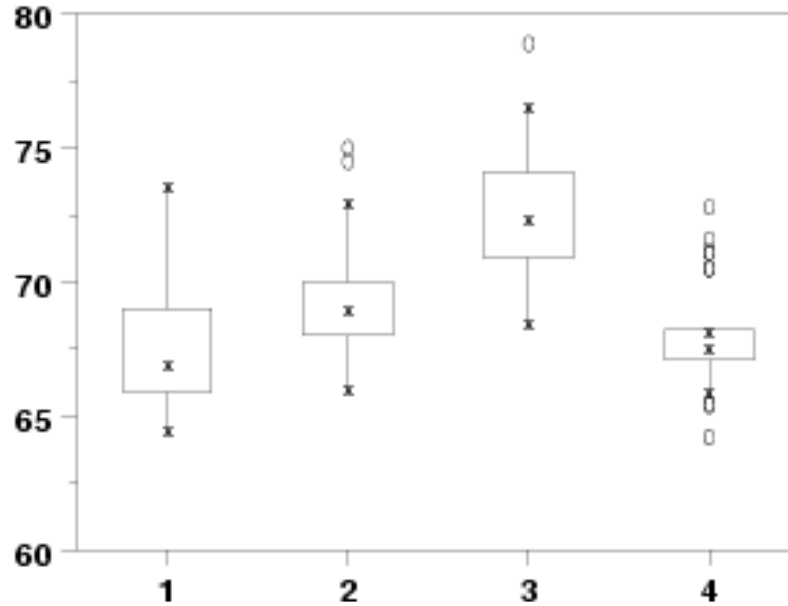
Outlier, trend and cluster analyses are discussed separately in the following two sub-sections.

4.4.2.1. Outlier Analysis

Tukey (1977) introduced the boxplot as a robust means of studying the distribution of a variable and for identifying extreme values and outliers. It involves a three steps process:

1. find the median, the lower quartile and the upper quartile of the data;
2. denote the median and draw a box around it between the lower and upper quartiles (marking the interquartile range);
3. lines (stems) are drawn from the upper and lower quartiles respectively to 1.5 times the interquartile range or to the maximum and minimum points respectively, whichever is less.

An outlier is a value that lies outside 1.5 times the interquartile range as illustrated in Figure 4.3.



Legend: Symbols 0 represent the outliers, while X represent lower quartile, higher quartile and median
Source: Chambers *et al.* (1983)

Figure 4.3. Box Plot

Another technique for outlier detection is Grubbs' test that is define in Grubbs (1969) and improved in Stefansky (1972). The test is especially effective in detecting outliers on normal distributed data. By using Grubbs' test any outliers can be detected and removed from the overall data. The process is repeated iteratively until no outliers are detected. Due to this iterative nature of this test the sample size of the data has to be high (over 6 points) for it to be valid. It is expressed in the following formula:

$$G = \frac{\max|V_i - M|}{\sigma} \quad (4.8)$$

...where M is the sample mean and σ the standard deviation while V_i is the variable being tested. Thus any point having the largest absolute deviation from the sample mean in units of the sample standard deviation will be tested as possible outlier against the confidence

limits of a t-distribution at N-2 degrees of freedom. Thus the hypothesis of no outliers by Grubbs' test can be rejected when the following condition is satisfied:

$$G > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{(\alpha/(2N), N-2)}}{N-2+t^2_{(\alpha/(2N), N-2)}}} \quad (4.9)$$

.....where t is the critical value of the t-distribution and N is the number of values.

This concept can also be adopted for a one-tailed test, allowing the user to find the outlier on the lower and upper extents. The formulas for outlier detection on lower and upper extents respectively using the Grubbs' test are:

$$G = \frac{M - V_{\min}}{\sigma} \quad \dots \text{where } V_{\min} \text{ is the minimum value} \quad (4.10)$$

$$G = \frac{V_{\max} - M}{\sigma} \quad \dots \text{where } V_{\max} \text{ is the maximum value} \quad (4.11)$$

These techniques are known as outlier detection which consequently leads to outlier removal.

A scatter plot can be used to define outliers for a bivariate description (Isaaks and Srivastava, 1989). To find erroneous data, data is plotted against a graph to ascertain those points which are completely out of harmony with the rest. These points which are out of harmony are best reviewed before being completely removed, as they might not be erroneous data but simply very uneven points on the plane (\Re). A scatter plot reveals

relationships or associations between two variables (Chambers *et al.*, 1983). Such relationships manifest themselves by any non-random structure in the plot.

A scatter plot is a plot of the values of Y versus the corresponding values of X :

- Vertical axis: variable Y which is usually the dependent variable
- Horizontal axis: variable X which is usually an independent variable

There is normally a problem that many analysts try to avoid during data polishing which is over polishing the data. Over polishing the data could actually serve as a negative point rather than positive one by actually hiding the important information. It might also cause the importance of a phenomenon to be diminished by overly smoothing the data. Therefore it is important to bear in mind this issue when processing data in preparation to modelling the variogram.

4.4.2.2 *Trend Analysis*

Spatial patterns are caused by trends on the plane (\Re) surface. Data pattern is an important consideration when modelling. Researchers in geostatistics refer to spatial and attribute patterns as part of the same issue. However even though there is a clear link between the two, here we identify their difference and the importance of that difference in building a better model.

The data patterns in geostatistics can be broadly defined in two aspects:

- Attribute patterns
- Spatial patterns

Attribute patterns are mainly concerned with non-linearity of the attributes under study. This can be seen when a planar graph has been plotted against the attributes of the spatial data to determine the offsets. In general such patterns are associated with regional trends in the phenomenon under study (Diggle, 1975; Cressie, 1993). In such cases, how the surface is configured becomes the main focus of interest. These patterns are comprised of the actual geometric arrangement, the surface trend and the presence of physical geographical features such as the topography. Whilst each attribute under study may have its individual trend, some of the attributes may be highly correlated and could be important for the analysis. In modelling a variogram, it is often the case that trends are removed prior to calculating the semivariance.

Spatial patterns concern the distribution of the x,y points themselves, irrespective of their attributes. Where sampling has been carried out to collect the spatial data, the distribution of the sampling points themselves may be random, on a grid or according to some other scheme. However, where the data are point events, that is, the points represent the occurrence of an event such as the presence of a tree or the occurrence of a crime, then the spatial distribution of the points can have structures that can be revealed through the use of the variogram (Ripley, 1981). This then is concerned with neighbourhood effects. This type of data is referred to as spatial point pattern by Diggle (1975).

Geostatistical data (point observations of a continuously varying quantity over a region) often has a trend in it. A trend is the non-uniformity of the surface where the geostatistics are to be operated. The trend often affects the outcome of the prediction and thus becomes one of the common issues in geostatistical data analysis (Isaaks and Srivastava, 1989). Thus, before a variogram is plotted, it is often important to remove this trend (large scale variation) using various techniques, some of which will now be discussed. There are many techniques available from the published literature. These include:

- **Polynomial regression:** one of the much used polynomial regressions is the multidimensional generalisation of Student (1914), which is currently referred to as a smoothing technique and is based on fitting the data (for a two dimensional surface) by the least square (Ripley, 1981), using the following general formula:

$$f(x, y) = \sum_{r+s \leq p} \alpha_{rs} x^r y^s \quad (4.12)$$

....where the integer p is the order of the trend surface and x and y determine the coordinates use for the regression process. By expanding and adding additional paremeters to the above formula, Cressie (1993) showed an example of a more complex trend detection formula, a quadratic trend. He defined this with the following formula:

$$f(x, y) = \alpha_{00} + \alpha_{10}x + \alpha_{01}y + \alpha_{20}x^2 + \alpha_{11}xy + \alpha_{02}y^2 \quad (4.13)$$

Thus, the given regression formula can be expanded to fit any size of trend.

- **Tessellation and triangulation:** this technique works by creating splines between data points (Ripley, 1981). There are two structures that can be attributed to this technology:
 - The first structure is known as the Drichtel Cell or Theissen Polygons, defined by dividing two points with a line (right in the middle) and thus creating continuous joining lines that form lattices (or tiles) on the plane (\mathfrak{R}) under study.
 - The Delaunay Triangulation defines two points which are near that could join up with a third to form a triangle. The structure works so that

triangulation creates a continual curve through a specified plane (\mathfrak{R}). For this structure, the work of Lawson (1977) is normally attributed to when deciding the joining line for measuring the angle (by choosing the larger of the two angles formed by the different possible triangles).

A weighted median polish is achieved by overlaying a grid (normally rectangular) on the plane (\mathfrak{R}) under study, picking appropriate interval size and placing nodes on those intervals. Cressie (1993) used 20 miles with 9 North-South and 24 East-West intervals.

4.4.3 Declustering

Whilst the ideal is to have a randomly distributed sample of data points, this may not occur in practice due to the practicalities of sample collection and particularly so if the data set represents point events. This may result in some areas being over sampled/represented and may introduce bias into any analysis, that is, the data may not be representative of the population. There is a consequent need to remove this effect through declustering. There are two broad approaches to declustering: count-based weighting and area-based weighting (de Smith et al., 2007). At its simplest, count-based weighting involves superimposing a regular grid and those cells which have an above threshold number of data points can be regarded as clustered or over-sampled. This threshold may relate to percentiles within the distribution. Two strategies can be adopted for declustering given the evidence of clustering, one being to randomly sample the over-sampled cells, the other being to weight all the points in the data set by the reciprocal of the count of points in the cell to which they belong. One problem in this approach is in deciding the appropriate size and positioning (even orientation) of the grid mesh over the data set. Area-based declustering is similar in that Voronoi regions are created around each data point and then weighted in proportion to the area of its Voronoi polygon. The weight can be calculated as

the area of each Voronoi polygon divided by the average size of all Voronoi polygons thus giving smaller weights to smaller Voronoi polygons that result from data points being closer together. Problems arise in this approach from edge conditions where the outer data points can have very large Voronoi polygons. Suggested solutions are to have a minimum bounding rectangle (MBR) or a convex hull.

4.5 Cross Validation and Goodness of Fit

Variogram validity can be measured using a number of mathematical tools. However, one well used tool is cross validation, which is a function achieved by kriging each sampled location with all of the other samples in the search neighbourhood, then comparing the estimated values with the true sample values. The technique is used to make a decision to use a variogram model over other models, which is decided according to their ability for making predictions. One cross validation process, known as 'leaving-one-out', takes the following steps:

1. Compute an experimental variogram of the sample data.
2. Fit all possible models (the experienced user needs to define the models that seem plausible to the content).
3. Estimate Z from the data for each model, using kriging at each sampling point.
4. Calculate the results using one of the available formulae. An example is the mean deviation or mean error, ME, as given below:

$$ME = \frac{1}{N} \sum_{i=1}^N (z(x_i) - \hat{z}(x_i)) \quad (4.14)$$

Normally this model will be completed through kriging to determine which of the possible models performs. Thus the ME result should be close to 0, since kriging is an unbiased prediction method.

Goodness-of-fit is an approach to measuring the fit of a variogram model to the graph of semivariences. This is done after the data have been modelled as a variogram. Looking into how good is the fit and thus how well the regression line fits the data is very important. The goodness-of-fit can be defined as the degree to which observed data coincide with theoretical expectations (Diggle, 1975). The test is based on a statistical test. One approach is to calculate the R^2 of the fitted model and the overall significance of the regression model using F test. Another approach is to use indicative goodness-of-fit as follows (Landim, 2004; Deutsch and Journel, 1998):

$$I = \frac{1}{N} \sum_{k=1}^N \sum_{i=0}^{n(k)} \frac{P(i)/\sum_{i=0}^{n(k)} P(i)}{h(i)/h_{max}(k)} \left[\frac{\gamma(i) - \gamma^{\downarrow}(i)}{\sigma^2} \right]^2 \quad (4.15)$$

.....where N is the number of directional variograms, $n(k)$ is number of lags for the k th directional variogram and i is the lag. $P(i)$ is the number of pairs in the lag and $h(i)$ is the mean distance between members of pairs with $h_{max}(k)$ as the maximum distance for the k th variogram. $\gamma(i)$ being the value of experimental variogram for lag i effectively $\gamma^{\downarrow}(i)$ is the value of the model variogram for lag i . The variance of the variogram estimation is denoted by σ^2 .

The fitness is considered good if the value of the criteria (I) is close to zero. This model can be achieved using having the lag distances, min and max distance and the number of pairs being computed for each lag.

4.6 Kriging

The term kriging is derived from Daniel Krige (1951), a mining engineer, even though the technique was described and made famous 13 years later by Georges Matheron (1965). These methods are now widely used in the minerals industry and have disseminated out into many other fields where spatial data is studied. In geostatistics, prediction is normally done using the data structure as represented by the variogram model which is used in kriging to determine an optimal set of weights to estimate the value of any unsampled location (Cressie, 1995). Thus, kriging can be used to construct contour maps, but unlike many other contouring algorithms it can provide a measure of the uncertainty of the contoured surface.

Ordinary Kriging uses a weighted average of neighbouring samples to estimate the 'unknown' value at a given location. The relation of Ordinary Kriging and variogram model is the use of later to optimize the weights and provide the relation between known and unknown values through the location of the samples. A "standard error" which quantifies confidence levels also provides the Ordinary Kriging. It is defined by two formulae:

$$\text{Predictor assumption: } \hat{Z}(r_0) = \sum_{i=1}^N \gamma_i Zr_i$$

$$\text{Model assumption: } Z(r) = \mu + \varepsilon(r) \quad (4.17)$$

Ordinary Kriging assumes that local means are not necessarily closely related to the population mean. For this reason it uses only the samples in the local neighbourhood for the estimate. Ordinary Kriging is one of the most commonly used kriging method for environmental situations. The Ordinary Kriging satisfies the B.L.U.E (Best Linier Unbiased Estimator) model. This model determines that by minimizing the variance of the errors and

dealing with weighted linear combinations of the data and making mean error equals to zero the prediction is more likely to be accurate. This makes Ordinary Kriging a very common technique for interpolation in geostatistics. For this point Cressie (1991: p119) has referred to it as “optimal predictor”.

The goal of Ordinary Kriging is defined by Isaaks and Srivastava (1989) as “ambitious ones and, in a practical sense, unattainable since MR and σ_R^2 are always unknown.” They managed to attain it as their calculations for the mean error and the error variance were possible through the exhaustive data set. However, they explain its usefulness by building a model of the data being studied and work with the average error and the error variance for the model. Thus a probability model should be used where both, the bias and the error variance can be calculated.

4.7 Conclusion

The purpose of this Chapter has been to establish the basic techniques and functions that are used in geostatistics. The aim here is to establish the techniques that in this research form the algorithmic basis of the software agents used to investigate distributed component GIS. The next Chapter goes on to investigate the various methodologies of developing and testing software agents.

CHAPTER FIVE: METHODS FOR IMPLEMENTING AND TEST DESIGN OF AN AGENT-BASED VARIOGRAM MODELLER

5.1 Introduction

This chapter sets out and explains the methodology for agent development for modelling the variogram and investigates its performance as a programming environment. The limitations of the methodology used for implementation will be discussed, and a proposed solution presented. The appropriate architecture for Agent Programming Languages (APLs) will be examined, as well as the novelty of their structure and implementation design.

5.2 Agent Development Methodology

Brazier *et al.* (2002) explain that methodology is very important for conceptual design. However, careful observation is required, as this is the stage where the knowledge-based techniques can be exploited and developed within known engineering which, in turn, will enable the design of more complex agents. Considerable research has been conducted on agent development methodologies (architecture) and, as a result, there is a range of methodologies available, each with its strengths and weaknesses. Presented here is the methodology focused on in this research, known as 'Tropos', as it covers many of the required research areas for agent modelling. The methodology for Tropos will be explored; its strengths and weaknesses will be compared with other methodologies (also on Appendix E).

5.2.1 Tropos

Tropos is an agent development methodology architecture which is designed to complement the flexibility of agent-based software during the development cycle (Bresciani *et al.*, 2000; Giunchiglia and Mylopoulos, 2000; DeLoach *et al.*, 2009). It provides a facility to capture the ability of agent behaviour and to satisfy the desired

functionality throughout the development cycle, such that knowledge functionality is considered during the modelling iteration of each development stage.

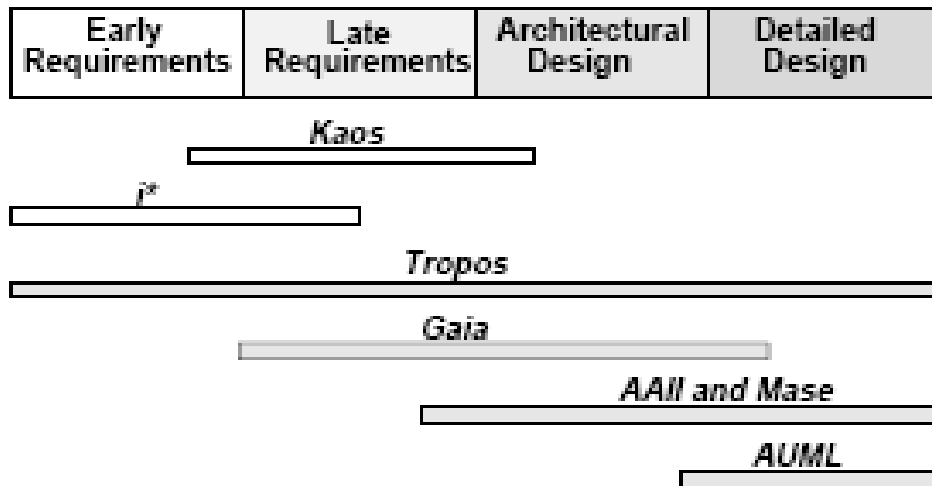
Development stages are clearly defined within the Tropos methodology, so that it is easy to start the development of an agent-based system. Tropos captures the system approach, individual agent approach and internal structure of each agent in the system, from system analysis to implementation. Five stages are specified in support of the analysis, design and implementation (Braciani et al., 2004), which are as follows:

1. **Early requirements:** like the object-oriented software requirement analysis stage, it captures the requirements of the system to be implemented. It specifies and defines preliminary system requirements and draws diagrammatic interpretations to facilitate the next stage.
2. **Late requirements:** captures the system requirements integration to its environment, such that by this stage the main agents are developed. At this stage, agents are introduced to their environment and are assessed as to their functionality.
3. **Architectural design:** this is where the system is defined, with its external actor's interconnection mechanism using data and goal flow. This stage is divided into three sub-stages:
 - The system's overall architecture; so it is not the agent but the system is defined from an architectural point of view.
 - Capturing the capabilities required to fulfil the system's goal (i.e. the collective goal of the agents).
 - Defining agent types and their capabilities which are required for full dependency.
4. **Detail design:** this stage specifies agent capabilities and interactions using Belief-Desire-Intention (BDI) structure. During this stage, action and plan

diagrams are established. These are identical to the action and sequence diagrams of the object-oriented design stage.

5. **Implementation:** this specifies how to implement the system at hand into the object-oriented agent development APIs. Tropos is mainly structured to work with JACK (Georgini *et al.*, 2001), but the structure is flexible and could be applied to other tools like JADE and ABLE.

Tropos has drawn upon many previously defined methodologies (like the Gaia and Yu i* paradigms) to enrich its functionality and increase flexibility (Bresciani *et al.*, 2000). Figure 5.1 illustrates a comparison of Tropos to other methodologies in relation to development stages. One limitation of Tropos methodology is that, during implementation, the programmer needs to revert back into Unified Modelling Language (UML) or Agent Unified Modelling Language (AUML) to convert the preliminary design and detail design into an object-oriented based design, to be implemented using Java (Giunchiglia *et al.*, 2002). To ease this process, the Tropos designers suggest the use of JACK API to implement the agent. Although using JACK minimises the amount of coding for a programmer, it does not simplify the implementation design, as object-oriented class diagrams, sequence diagrams and activity diagrams need to be provided to allow the programmer to code agents. However, the explicit representation of the goal and plans of an agent, and the direct reference to BDI structure allows flexibility in the run time adjustment to deal with unforeseen circumstances (Giunchiglia *et al.*, 2002). This feature is not available in pure OOP.



Source: Braciani *et al.* (2000).

Figure 5.1: Comparison of Agent Methodologies and Functionality

Mylopoulos *et al.* (2002) point out the extensive capabilities of Tropos, by fitting organisational structures (such as auction structures and joint venturing) to show its strength and flexibility. Because of the many successes when using Tropos in real life (like the ice producing company project identified by Garzetti *et al.*, 2004) and its ability to capture every step of a development cycle, it can be viewed as the complete architecture for intelligent agents. Tropos would be utilised for the development and proposed programming language structure that can be directly applied to the implementation level of the methodology to create an agent variogram modeller.

Even with all these advantages, Tropos still requires improvement. As it currently stands, there is no available agent-oriented architecture which is clearcut for use from design to implementation. All the available methodologies (including Tropos) only provide a transition to UML class and other relevant diagrams (Wooldridge *et al.*, 2000; Giunchiglia *et al.*, 2002; Mylopoulos *et al.*, 2002; Garzetti *et al.*, 2004). Tropos has a tool to by pass this stage by auto generating the code using Taom4E but this process does not allow the programmer to understand the code for maintenance. This lack of suitable architecture increases the required architecture complexity and lengthens the

development cycle, and creates a bottleneck at the implementation stage. This is the principal reason why some agent software ends up being considered as a conceptual design but is not then implemented. Even so, object-oriented software development also has many methodologies (Stevens and Pooley, 2001). These object-oriented methodologies are themselves moving toward rapid application development to secure software integrity and thus reduce development time.

It is sometimes argued that agent software development is only useful in specialised situations, but historically object-oriented software initially started out for use in specialised cases (Dahl *et al.*, 1972; Dijkstra, 1994; Bray, 1997; Potok *et al.*, 1999), much as agent-oriented software development is today. Nevertheless the usefulness of the architectures developed has led to their becoming the foremost software development structure currently. This trend is now moving towards agent-oriented software development.

5.2.2 From Design to Implementation

It is necessary to clearly define when agent design stops and implementation starts. It can be where the rapid application development structure of methodology refers back and forth at the development stage, but a programmer needs to be told when to start and how to write the code. So a structure is needed to cut out the object-oriented UML transition in agent-oriented software development. This would help separate the work tasks of the system analyst, designer and programmer, as current architecture proposes that these three stages (or jobs) run simultaneously throughout the development cycle, without any clear cut agreement of where to start and where to stop. Albeit that this problem seems less obvious nowadays, as we move into the era where agent software is becoming the norm, programmers will start taking the initiative to build large software with this style of architecture which will become cumbersome and hard to adopt.

5.3 The Programming Environment

Abelson and Sussman (1984: p12) state that:

“First, we want to establish the idea that a computer language is not just a way of getting a computer to perform operations but rather that it is a novel formal medium for expressing ideas about methodology. Thus, programs must be written for people to read, and only incidentally for machines to execute.”

It often occurs that, during the process of new software paradigm development, the emphasis of software development shifts completely toward methodology and how the software is to be structured, while the focus on how to implement it only comes much later in the process. Programs or discreet variables are essential to computer software and so a strong programming language structure would thus ease the process of implementation. This is simple cause and effect. Tools for developing agents have not yet been formalised, although considerable research is currently being undertaken to standardise this issue (Travers, 1996; Voyles, 1997; Odell and Bock, 1999; Zhao and Jo, 2003; Jo and Einhorn, 2005; Yu et. el. 2007; Sterling and Taveter 2009).

Arguably, development methodology of agent-oriented programming is also improving at a very fast pace, but implementation still relies upon object-oriented languages like Java and C++ (Nute *et al.*, 2004). For example, a number of these tools have been considered during the design of the multi-agent platform ‘Gulliver’s Genie’ (O’Grady and O’Hare, 2004)(such as FIPA-OS, LEAP, Zeus and JACK). These tools are described as a new area of software engineering, entitled agent-oriented software engineering (AOSE) (O’Grady and O’Hare, 2004).

As the Java development environment is both platform independent and object-oriented, it is often recommended to be used as an integral programming language for the agent applications (van Breemen and de Vries, 2001). Even if agents are to function cross-platform (as heterogeneous systems) on a network environment, there will be no conflict between the client and the server, or from one peer to another (Shih *et al.*, 2001). Furthermore, Amandi *et al.* (2005) have specified an agent development language,

JavaLog, which is based on the Java object-oriented environment, tapping into Prolog to deal directly with the intelligence element (i.e. the required knowledge) of an agent system.

With regard to the network and mobility of an agent, a communication platform must be established to allow the agents to communicate with each other. This can be achieved by using an inter-agent communication language (such as FIPA's agent communication language) that was used to develop the 'Gulliver's Genie' (Farjami *et al.*, 2000), DESIRE model (Brazier *et al.*, 2002) and Petri net (Kwon and Lee, 2001). More importantly, agent development must consider and, in broad terms, allow modularity.

KADS is a methodology, presented by Coffey (2003), which is solely implemented to make sure that modularity is achieved during agent development. This methodology is derived from a group of models, each of which represents some part of the agent system (Schreiber *et al.*, 1994). Furthermore, van Breemen and de Vries (2001: p247) have determined that an agent-based system "should be decomposed into a set of supposedly independent control tasks". On the other hand, simulations have paved the way for experiments. When there are no concrete mathematical models of the system and also the increased possibility of system damage during such experimentation, simulations are probably the best approach to represent the actual behaviour of the system and so minimise the risks of real system damage (Lucey, 2002).

One of the main differences between the simulator and the experiment is as follows. A simulation of a small section of the system may be developed then, modularly, extra components can be added as required or desired to represent the system (Nute *et al.*, 1995). In comparison, experiments engage the whole available system (whatever modules are under test, whether being the whole system or just the part being tested) into 'try and test' mode. Simulators could even be equipped with an interface that assists the user to deal with component interaction, extension and augmentation issues (Crookston, 1997).

Once the agent has been developed and simulated, its unknown performance still remains to be tested at the deployment level of the development cycle. Performance can be measured using an experimental approach. A scenario can be established. A practical example is seen in Lee *et al.* (2003), quantitatively experimenting their online auction application 'MoCAAS' (mobile collaborative auction agent system), where they set an interval of thirty minutes for six hours to establish the network load when using the application with or without agent employment. Likewise, van Breemen and de Vries (2001) had previously used a similar experimental approach when they developed a tracking controller-agent to complement the design and implementation of a domestic room thermostat, with an experimental tool to objectively compare two house temperatures for a period of sixty hours. The experiment was to establish the key value that should decide the proposed agent framework. Here, the experiment was used as a tool to empirically initialise the deduction of data through a positivist approach, which consequentially led to explanations and predictions (Blaxter *et al.*, 2002) of the suitability of function for a tested framework.

From the above published studies on programming environment for agent development, it has become apparent that in order to achieve a more rapid and constructive way of developing an agent application, a formalised and most appropriate methodology is needed to be established. Another point identified from the literature is that certain methodologies are more likely to yield successful results than others. The argued implementation methodologies and appropriate methods should have the following features:

- modelling languages – such as UML;
- a modular approach development model;
- object-orientated;
- logic-oriented.

As discussed, Amandi *et al.* (2005) has suggested the use of JavaLog for implementing agents, which has the ability to allow modularity without compensating knowledge

behaviour. However, for many mainstream agents, problems have arisen with the implementation programming of APIs into Java and C++.

Any proposed structure of agent-oriented programming should map the development of OOP. OOP provides a good example of the evolutionary process of programming development. It started during the early 1960s, when Nygaard and Dahl set out to develop *Simula 67* (Bray, 1997). Their prime objective was to develop an easy structure to implement objects, as described through object-oriented development design. During these early years, it was perfectly normal to implement objects using the widely available structured languages such as C and PASCAL (Sutherland, 1999). Due to the growth in popularity of OOP, many languages have arrived on the scene since the 1960s. However, one can still code objects through structured language, except the programmer will need to be highly knowledgeable in both structured language and the object-oriented design in question. This makes it time and resource consuming, neither of which are in abundance in today's world.

However, mapping the development of OOP should utilise the capabilities of OOP itself to easily build objects, and not use OOP to build software that are to be called 'agents'. A new programming language will be superior to another if it is easier to implement, more secure and functional. According to Bigus and Bigus (2003), the race towards producing such a language structure for agents started around the mid-1990s, and they have compiled an extensive list of attempted platforms and APIs for agent development. This list ranges from agents that focus on mobility rather than intelligence (e.g. FIPA-OS), to agents that focus on intelligence rather than mobility (e.g. AgentBuilder), including everything in-between (e.g. Aglets, JADE and JAKE). However, most of these suggested and enforced languages lack functional requirement features and need to use OOP to achieve their agent development goal. These current agent languages are based on the API structure, imported into the Java language. The agent Java APIs are very similar, with different implementations or sometimes minor functionality.

Considering a few, JACK, JAM and JADE are each very similar in all development aspects. Such similarities are:

1. *They are all object based and possess super classes:*

- *behaviour*
- *agent*
- *goal*
- *action.*

These classes are simply object-oriented super classes that need to be inherited and mostly rewritten in the OOP structure to provide agent software. This is often cumbersome and the programmer needs to be very efficient in OOP before they can start coding the simplest agent. Looking at object-oriented programming during its early years, the main focus emphasised that it offered flexibility to implement (through easy coding), with better programme control and software security developed through encapsulation.

2. *They all are APIs running on the Java Virtual Machine, superseded by Java programming language*

The difference between JACK and other agent development APIs is that JACK offers the visualisation of agents during the development of the system (Bresciani *et al.*, 2004). Furthermore, all languages (including JACK and JADE) are object-oriented-based, requiring extensive knowledge of object-oriented programming for implementation.

The direct implications of evolving object-oriented into agent-oriented programming can be seen by the work of Giorgini and Henderson-Sellers (2005), who focus on showing the influences of object-oriented programming on agent-oriented programming. The issue, though, to reduce complexity in agent-oriented programming, rather than

agents has been emphasised by Wooldridge (2003), and a new language based on reaction needs to be produced.

Reactions would allow agents to react to changes in the environment. Thus, an agent would be able to perform a function (or action) and, if while executing it the environment changes, then it should change the action accordingly. This is not possible utilising the currently available programming languages, as one function has to terminate before another executes. Thus, reactions should allow a subsequent change of action without terminating the previous action. This situation occurs when the environment is too complex: “In domains that are too complex for an agent (or MAS) to observe completely or there are uncertainty in the environment (the agent cannot assume its reason for executing is valid) the assumptions are not reasonable” (Wooldridge, 2002: p9). This statement shows that, in most situations, when the environment changes and the agent executes an action to reach the agent’s goal according to the new environment, it should still observe its belief of the environment. If this belief changes it should react to this change, but still execute the previous action in the background without changing the state.

5.3.1 A Proposed Agent Programming Language Structure – ‘Agent Diagram’

In this Section, a structure for agent programming language, named as ‘agent diagram’ is proposed, see Figure 5.3. Class diagram concept is used. The class diagram shows how an object behaves, and what data and functions it encapsulates. All other functionality of an object is drawn in other interaction diagrams, such as the state diagram. Using class diagram can expedite object-oriented software development and improve communication between designers and programmers. A similar structure is required for agent-oriented software development.

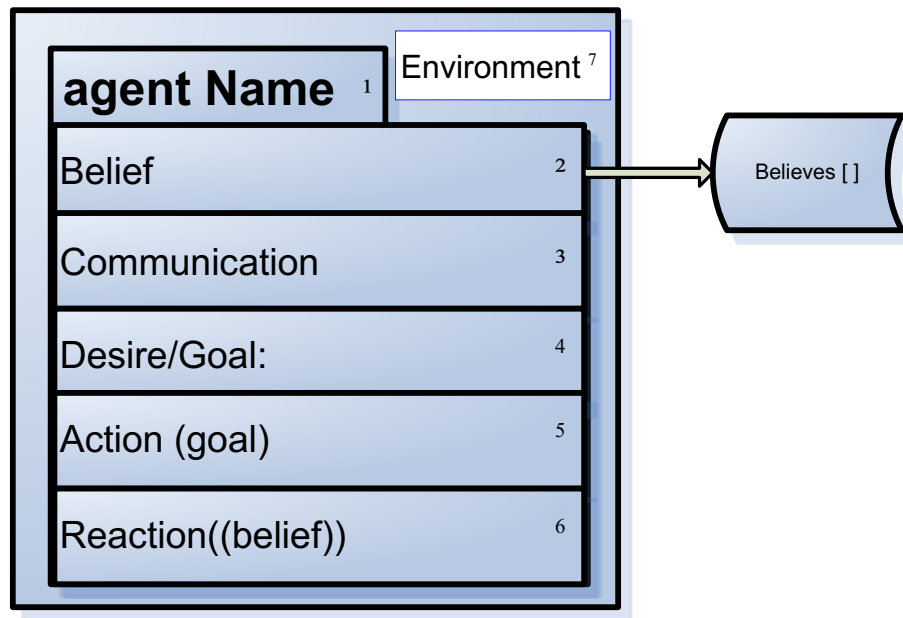


Figure 5.3: Agent Diagram - Structure for a Top View of Agent Representation

Figure 5.3 demonstrate the concept of the programming structure of a proposed agent. This structure can be supported by the JADE or JACK programming syntax. This 'agent diagram' provides information on:

1. **The agent name** – required in order to be identified by other agents; such agents could be software, hardware, a human or any external factor, as discussed by Wooldridge and Jennings (1999) and Wooldridge (2003).
2. **Belief** - the status of the environment being sensed by the agent (Voyles, 1997; Zhao and Jo, 2003). Thus an agent might have a single belief (as shown in our implementation example) which could be a set of states, and which in turn are stored in an array type of structure that is present in OOP. This acts like the attribute element of the object-oriented class diagram. However, in the case of the agent, it is important that the data are not only accessible by specified functions, but also that it is actually mapped to goals that are achieved by actions taken. When the belief of the environment changes, it triggers a reaction that

rechecks the goal - if the status of the environment is such that the goal wants it to be changed (or needs to be changed), then it needs to generate an appropriate action.

3. **Communication** – is the mechanism that allows an agent to contact other agents. Here, the programmer can specify the type of communication and which other agents this agent can communicate with. In a hierarchical structure, an agent can have communication protocols established (e.g. I am superior to agent type X but inferior to agent type Y, so I can respond to orders from X but not from Y, and also I can give orders to Y but not to X). This is where the dependend-dependee structure is identified for the first time during implementation. The communication structure is defined as where agent A can communicate with agent B directly, or the communication is passed to the desire (or goal) level where agent B only needs some task done by agent A for it to continue on its state change mechanism
4. **Desire (or goal)** – what an agent wants to achieve in response to the environmental changes. Here, the goals that have to be achieved for the prescribed agent in order to satisfy the goals of other agents are identified, then linked to the agent in question at the appropriate level (e.g. the action of agent A that is required to trigger an action from agent B will be linked by an arrow directly to the name of agent B). Also, if a goal of agent A needs to be satisfied so that a goal of agent B can be satisfied, the arrow will link goal to goal. Using this section of the diagram, communication between objects can be shown (see Figure 5.4).

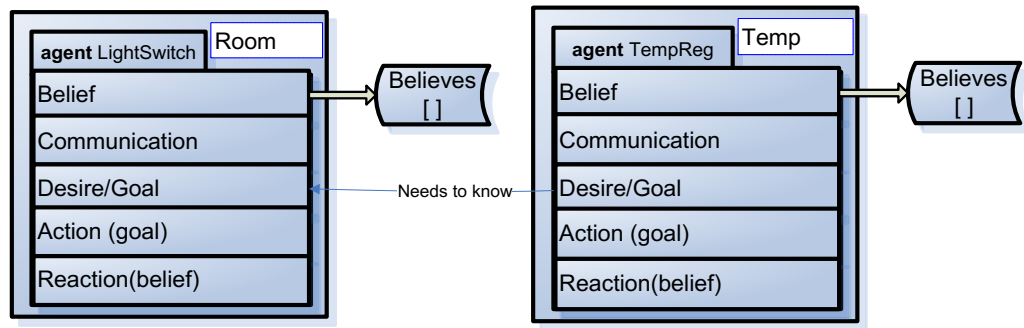


Figure 5.4: Connection Between Two Agent Diagrams, Showing Benefit

This structure is very comprehensive and supports the current analysis and design phases specified by Tropos (Bresciani *et al.*, 2004). Like the class diagram in object-oriented programming, here the number of goals that one agent requires another to satisfy can be annotated, as well as the type of relation which makes it easier for the programmer to implement. Figure 5.4 shows an example of the relation 'needs to know', which is that: the LightSwitch agent can only detect its environment and make sure there is light if the temperature agent returns the room temperature.

5. **Action (goal)** - the various ways an agent can execute an action to keep the environmental state as required. These are often referred to as 'capabilities' (Jo and Einhorn, 2005). Figure 5.5 provides an example of functions that could be executed to reach a goal.

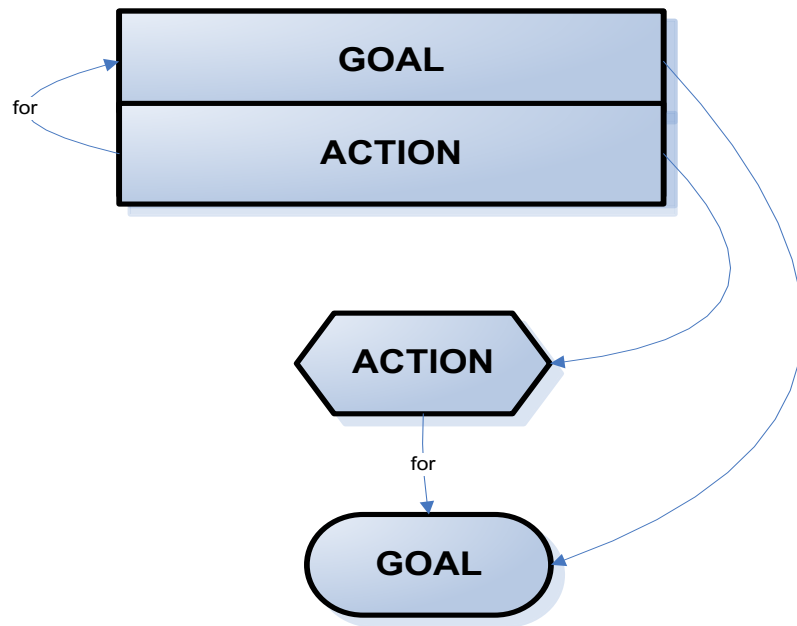


Figure 5.5: Depiction of Action for an Agent

6. **Reaction (belief)** –the structure used to monitor the environment and trigger an action according to change (as per belief). The reaction, like belief, constantly monitors the environment but, unlike the belief which can change simply by sensing the environment, the reaction will only occur when it gets confirmation of the environment change. There are major differences between action and reaction, the most important being that reaction is initiated by the environment and belief, while action only takes place to reach a goal (i.e. action is only triggered in order to reach the desired goal). This is illustrated in Figure 5.6.

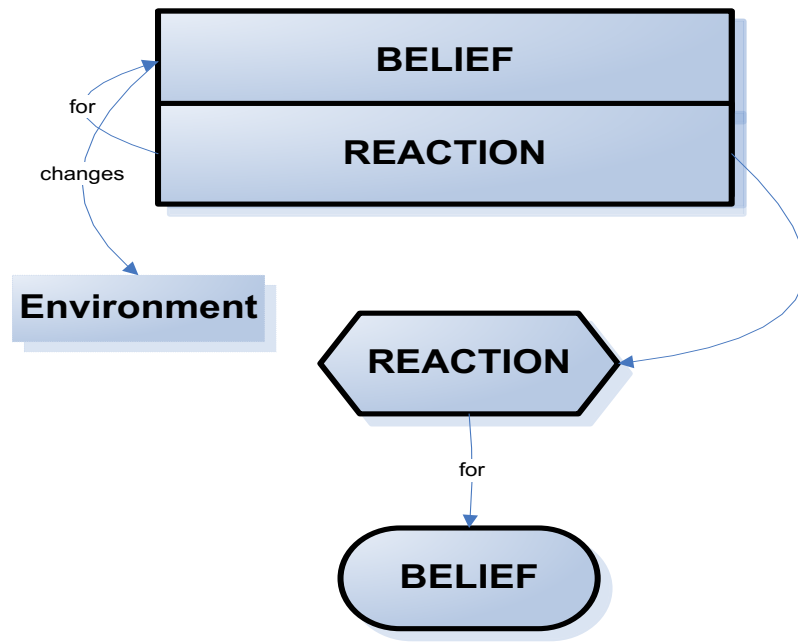
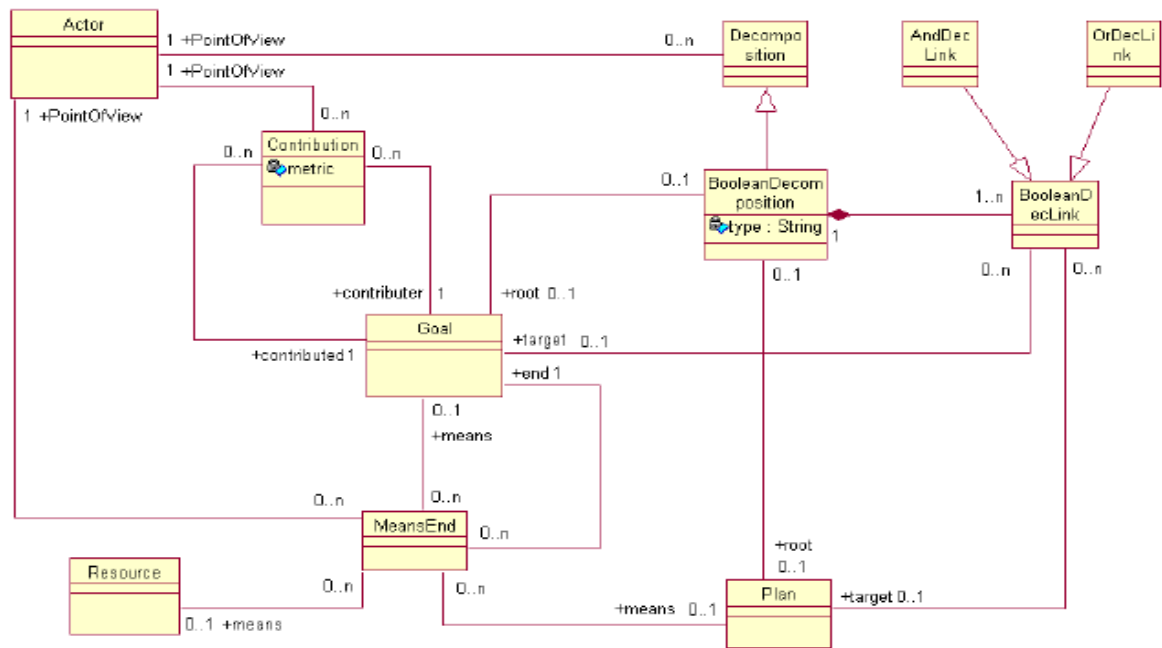


Figure 5.6: Depiction of Reaction by Agent

7. **Environment** - this has to be close to the belief. This could be anything that an agent supports or observes. It could be a human, another agent, the agent itself, a software environment or a physical environment.

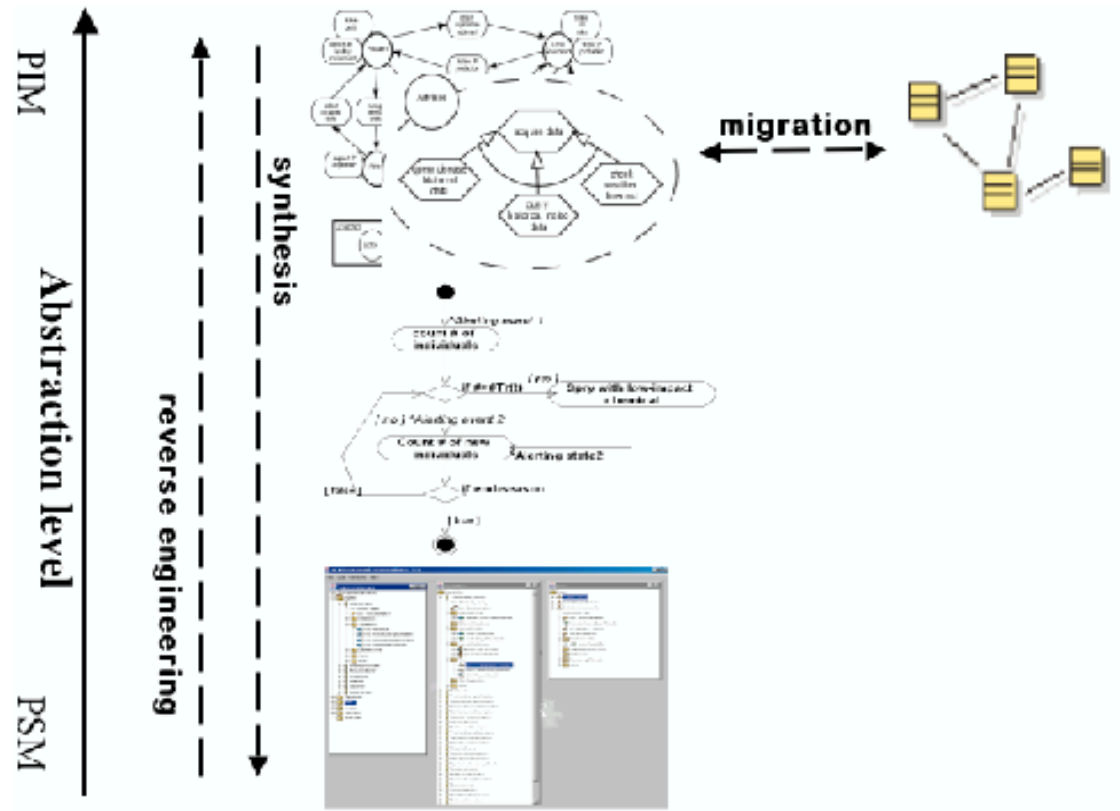
The 'agent diagram' structure (being for implementing agents, as compared to a class diagram for object-oriented implementation) could be compared to what Tropos currently specifies for the implementation stage. Figure 5.7 shows the general actor (agent) structure of the agent implementation diagram provided by Tropos (Perini and Susi, 2005).



Source: Perini and Susi (2005).

Figure 5.7: Structure of Agent Implementation

The structure suggested by Perini and Susi (2005) is necessary, because each attribute and action modelled for the agent needs to be a separate object which must be physically coded to fit the purpose of the agent. However, to implement this structure, a mapping of the agent-oriented detail design, which focuses on the specification of agents' capabilities (goals) and interactions, is needed. This is an unnecessary extra step (with added complexity) that has to be taken into account when building an agent-based system. Figure 5.8 shows the suggested mapping structure from Tropos methodology to object-oriented class provided by Perini and Susi (2005).



Source: Perini and Susi (2005).

Figure 5.8: Mapping Structure of Agent: Tropos Methodology to Object-Oriented Class

Figure 5.8 is more complex, yet it provides much less information for the programmer than the 'agent diagram' proposed in this thesis. Features like contribution (as the contribution to the system at large) are not necessary if each agent developed is (supposed to be) autonomous. Thus, they do what they are supposed to do, regardless of other agents, but only according to the environment (Wooldridge *et al.*, 1999).

Like the class diagram for object-oriented design, 'agent diagram' simply points out the components required for a single agent to function. In reality, this is the most important view for the programmer. It provides more information and less confusion.

Also in the 'agent diagram' structure the depender-dependum-dependee structure is explicit. Garzetti *et al.* (2002) introduced a structure to investigate dependency complexity and criticality for each depender-dependum-dependee structure. This structure is of immense help at the implementation level and even to simply understanding the system's performance. However, each agent should execute autonomously. Because of this, the dependee should only be conceptual for the programmer's information, such that one agent can change the environment so that another can act, but the second agent should only act when required by the changed environment.

Another important issue is that with the current Tropos structure there is support for modularity, but if any module (extra agent functionality) is added then the development cycle almost literally starts again from early requirements through to implementation (UML), where the object-oriented notion gets revised and modularity is achieved. However, with the 'agent diagram' structure, an agent could be added to the environment easier and quicker.

5.3.2 Agent Communication Infrastructure

Agents can communicate with each other regardless of their language structure. In this sense, the language structure refers to the implementation software of an agent. Suppose that one agent is developed using C++ and the other using Java. With the Multi-Agent System (MAS) structure, these two agents could be brought to an understanding (Wooldridge, 2003). Intelligent software agents can communicate via a model definition language (MDL) to integrate data with models. The MDL provides an inter-agent communication protocol for model development. Through the MDL structure, agents can communicate and thus retrieve, manipulate and store data from the available databases regardless of the type of data structure used. Agents can also request other agents to perform tasks. "To accomplish this task agents parse an MDL query and

translate tokens into a sequence of software specific spatial operations that transform the data into a form that is usable by particular models” (Bennett *et al.*, 1999: p153).

Using MDL as communication protocol described above, intelligent agents can match datasets (irrespective of vendor formats) by creating wrappers around, for instance, wrapper built around GIS data sets and possible modelling software. This is aided mainly by the functionality of ontology, as it expresses the ability of agents to act on their own without the need of interaction by other agents. Agents can simply react according to an action, despite the formats and styles which data sets employ. In other words, an agent can share data of different formats and styles. The concept of semantics is therefore emphasized. The basis of the Summary Schemas Model (SSM) as explained by Miller *et al.* (2001) suggests “a system taxonomy and map each local database attribute to a term in it”. An SSM could provide a way to unify database terms and their semantic meanings through implementing ontology. This is done to minimise the amount of data needed to represent large amounts of information (Miller *et al.*, 2001). “For imprecise queries, the first problem is to take query terms and map them to database terms. Therefore, minimally we must modify the ontology to make it database specific” (Lin *et al.*, 2001:2).

As an intelligent agent provides the ontology, the described structure could work by giving an agent the semantic of the other agent. If an agent encounters another agent with a semantic that it does not understand, it should then in theory request help from a mutually-understanding agent to acquire the semantics, or even to get the task done.

5.3.3 A Code Structure for Producing an Agent Programming Language

The most important issue on bridging the gap from the implementation of designs to coding is to have diagrams which can produce a basic structured code without worrying about the complexity of the performance code (Somerville, 2002). Such a structure would enable the rapid application development of the software. However, this structure

with the current topology of Tropos or any other methodology (e.g. Gaia) would have all the complex migration processes from agent-oriented design to object-oriented design, where the classes and function to build even a single agent are to be established. With an agent programming environment such as JACK or JADE, it requires a minimum of four inheritances and three objects to be developed (Bresciani *et al.*, 2004). When an agent system is created, it requires from one agent to many agents where each agent consists of minimum of four inheritances and three objects.

A programming language should have a structured template to which the programming environment or the integrated development environment could adapt. This structure can be seen in object-oriented flexibility (Java, C++), where APIs and integrated development environments are constantly evolving, aiming to reduce the complexity, increase the ease of coding and reduce the amount of developer coding by increasing the native code. By comparison, for structured language, if any changes are to be made on the language the whole previous platform becomes obsolete. Given below are basic guidelines for developing an agent programming language. It should have:

- a structured style of coding (uniformity);
- a minimised amount of coding for the programmer;
- robust coding;
- an easy way of achieving software improvement (modularity);
- security in the code (e.g. encapsulation in OOP); and
- security in the software produced (e.g. during software interaction).

These guidelines would facilitate the development of an agent programming language. From this perspective we can consider JACK or JADE. One can see that in object-oriented programming that Java is not the OOP language, but is an environment that offers OOP. There are many other environments that could directly be compared to Java, which include C++, C# and Delphi. All these environments have the same OOP structure, where each object is developed using the class notion. Each class has:

- a name;
- attributes; and
- functions.

These are always mapped to a diagram (called a class diagram) which shows the exact name of the object, its attributes and functions. A class diagram is shown in Figure 5.9 and the code template accompanying this class will be written as shown in Figure 5.10.

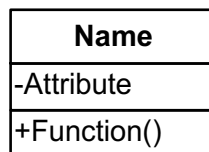


Figure 5.9: UML Class Diagram

```

class Name
{
    private Attribute;

    public Function()
    {
    }
}
  
```

Figure 5.10: Translation of a UML Class Diagram Into Object-oriented Program Code

Security in OOP is achieved through encapsulation, where only an object itself has the power to manipulate its data. Other objects can use this data via the owner object's functions. This makes the data secure, in that it cannot be changed by multiple objects and cause conflict in the system (Charatan and Kans, 2002). These functions are also displayed in the class diagram, which simplifies the understanding and coding complexity for programmers.

Such a structure is required for agent-oriented programming. To achieve this, a modelling structure is suggested here, and is put into a coding template. Figure 5.3 shows the 'agent diagram', while its accompanying code structure is shown in Figure 5.11.

```

agent Name
{
    belief
    clonable (true)
    {
        partial as -----
        {
            //structure of partial clone: possible renamed to reproduction
        }
        full as Name
        {
            this // this refer to exact copy
        }
    }

    communication
    {
    }

    goal :
    {
    }

    action (:)
    {
    }

    reaction (())
    {
    }
}

```

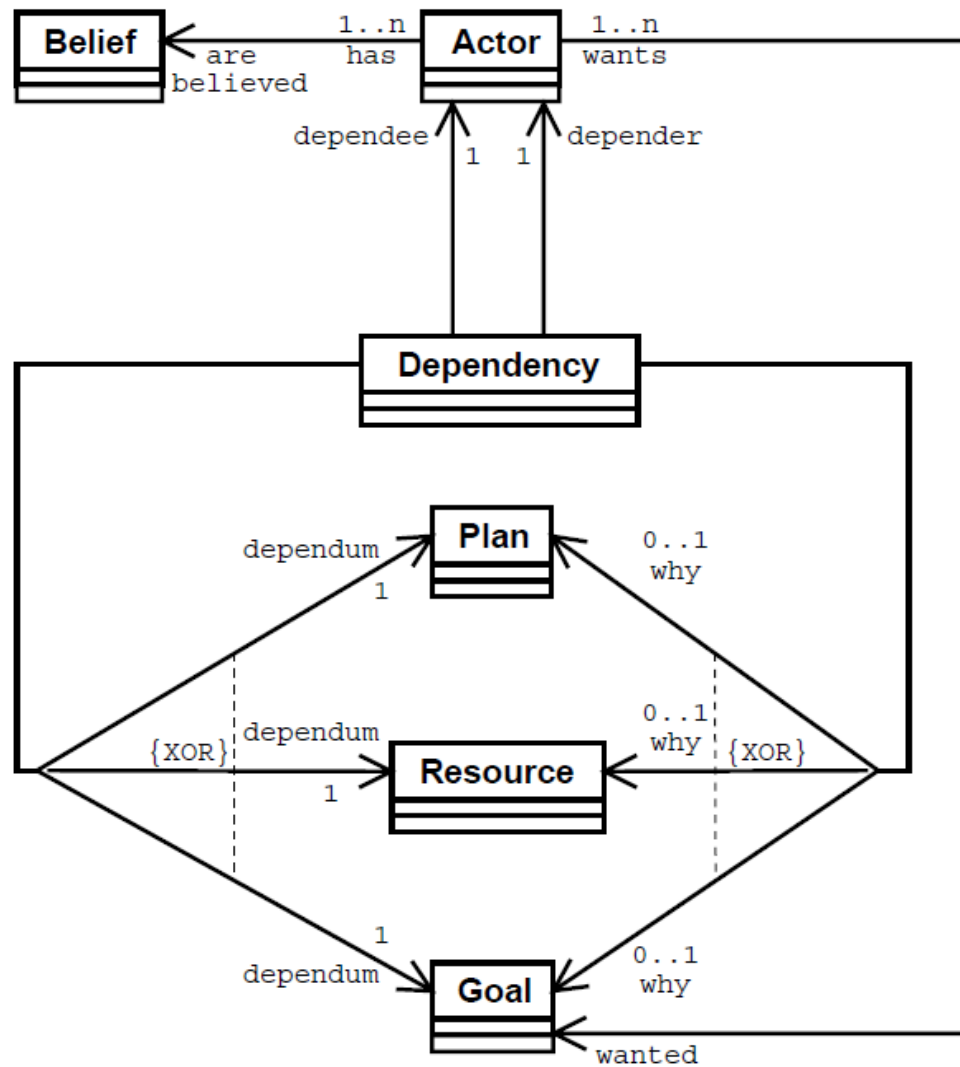
Figure 5.11: Code Structure for Agent Diagram

This code structure can also support dynamic learning, where an agent is able to add beliefs into its knowledge domain. Figure 5.11 shows only the initial beliefs that an agent, being instantiated knowledge, needs to have. The agent is provided with a database. This is identified as a vector structure that holds a list of beliefs. For programming, this database can be presented in many forms. For example, it can be presented like a notepad where each belief is written on its own line, or be an SQL supported structure like Microsoft Access DB. If Java Virtual Machine (JVM) is to be used as the supporting

execution environment, then the appropriate database system would be Java DB (provided by JVM). This structure would allow the newly acquired knowledge to be dynamically added into the agent's database. Furthermore, using a vector-like collection array will allow a faster knowledge search, resulting in real-time reactions and improved belief control and manipulation.

The proposed guidelines on code structure of agent programming language has improved utility in comparison with current implementations using JACK, JADE or other agent-based APIs. Figure 5.12 (based on Bresciani *et al.*, 2004) displays the implementation structure provided by JACK which is one of the widely used agent development tools. See Section 5.3.4 for description of JACK.

Shown in Figure 5.12 is also the implementation (code structure) for a single agent that is presented as an actor which has goals and beliefs. The depender-dependum-dependee structure can be seen in this figure, which is already presented in the agent-oriented detail design concept.



Source: Bresciani *et al.* (2004).

Figure 5.12: Actor that has Goals and Beliefs

Comparing the above to object-oriented design, where the complex structure of how the code should perform is left on the collaboration, sequence, state and activity diagrams (Pooley and Stevens, 2001). The proposed structure is built to save time and reduce the complexity of the initial coding of an agent (Somerville, 2002). Figure 5.13a represents the coding structure on the JACK platform, while Figures 5.13b and 5.13c show how each line of the code requires its own coding (increasing the complexity) and own class of coding respectively. Thus, has capability PresentQueryResults in Figure 5.13a is

expanded to Figure 5.13b, and #uses plan EvaluationQueryResults in Figure 5.13b is expanded to Figure 5.13c.

```
public agent UserInterface extends Agent
{
    #has capability GetQueryResults;
    #has capability ProvideUserSpecification;
    #has capability GetUserSpecification;
    #has capability PresentQueryResults;
    #handles event InformQueryResults;
    #handles event ResultsSet;
}
```

Figure 5.13a: Coding Structure of Agent on the JACK Platform

```
public capability PresentQueryResults extends Capability
{
    #handles external event InformQueryResults;
    #posts event ResultsSet;
    #posts event EmptyResultsSet;
    #private database QueryResults ();
    #private database ResultsModel ();
    #uses plan EvaluateQueryResults;
    #uses plan PresentEmptyResults;
    #uses plan PresentResults;
}
```

Figure 5.13b: Increased Complexity of Coding Structure on the JACK Platform

```
public plan EvaluateQueryResults extends Plan
{
    #handles event InformQueryResults ev;
    static boolean relevant (InformQueryResults ev)
    {
        return true
    }

    static model md;
    static queryResults qr;
    body ()
    {
        if (readQueryResults (qr))
        {
            if (findResultModel (qr,md))
            {
                if(compareResultModel(md))
                {
                    {storeResults(qr,md)}
                }
            }
            else storeEmptyResults();
        }
        else { System.err(1);
    }
}
```

Figure 5.13c: Coding a Sub-class Which Increases Complexity for Coding an Agent on the JACK Platform

The new code structure is similar to that suggested by Zhao and Jo (2003). They provide a more robust structure in their proposed language than is present in JACK, reducing much of the complexity of the coding and consequently requiring much less coding. Figure 5.14a represents an example of the structure of the programming environment being proposed here, while Figure 5.14b shows the accompanying code structure generated from it.

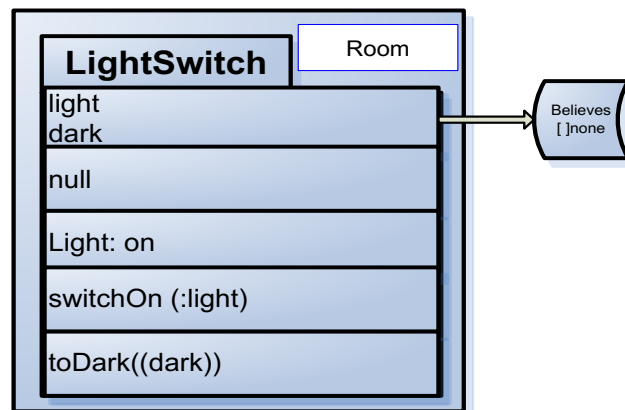


Figure 5.14a: Representation of the Structure for the Proposed Programming Environment

As can be seen, this code structure tightly ties an agent's design and code together and encapsulates an agent to its goals, desires and capabilities. In the time available for this research, it has not been possible to implement and validate a new agent programming language, but it was deemed important to define an appropriate language structure for easy agent development (at least at a design level). Thus, although the algorithm implementation of this thesis is based on this conceptual development and implementation mechanism, the actual implementation has had to be based upon code transfer and mapping to the JACK agent platform. Consequently, the variogram agent component (VAC) will be presented using the proposed design structure. This is intended to make it easier to understand the function of each agent, which should also help to make it easier to read for the lay user and for the future improvement of the system.

```

agent LightSwitch
{
    belief light = on
    belief dark = off
    clonable (true)
    {
        partial as -----
        {
            //structure of partial clone
            //possible renamed to reproduction
        }
        full as LightSwitch
        {
            this // this refer to exact copy
        }
    }
    communication
    {
        null
    }
    goal :light
    {
        switchOn (light)
    }
    action switchOn(:light)
    {
        light = on
    }
    reaction toDark((dark))
    {
        If (belief == dark)
        {
            switchOn(light)
        }
    }
}

```

Figure 5.14b: Code Generated from Figure 5.14a

This chapter so far has emphasised the importance of having the right agent platforms available when introducing agents into a software development environment within a discipline. The study of software development for agents identified that there was a deficiency in the ease of understanding methodologies for non-experts. Thus, two areas were identified that needed improvement:

1. the design stage of methodologies (specifically for diagrams that transit from design to implementation); and
2. the programming languages used.

For the first point, a new and more efficient diagram for software agent design has been proposed in the thesis. For programming languages it was determined that this required considerable additional studies, which were beyond the scope of this thesis. Thus, the focus of this study shifted to the environment. The best programming environment available was identified and utilised - this environment being JACK.

5.3.4 JACK Environment for Programming Agent

JACK platform is fully integrated software development engine. It used Java Virtual Machine (JVM) and has pre-constructed agent characteristics and their interrelation. For ease of use the engine developers have constructed a collection of autonomous agents which sense their environment and communicate with each other. Thus providing a simulated of encapsulation. Individual agents are defined using BDI concept which is translated to Capabilities, Plans, Events and Belief sets, designed to able to perform functions autonomously within its given environmental context. The JACK environment has extended programming syntax and semantics to allow ease of agent development (see. Fig5.15).

```
agent AgentType extends Agent {implements interface}
{
    // JAL declaration statements - the following declarations may be
    // used in an agent definition (when required).

    #[private,agent,global] data Type Name (arglist);

    // The agent handles events of type EventType.
    #handles event EventType;

    // The agent uses a plan of PlanType.
    #uses plan PlanType;

    // The round robin task manager is to be used - there are others.
    #uses taskManager SimpleRRTaskManager(steps);

    // The agent posts events of type EventType to itself.
    #posts event EventType reference;

    // The agent sends events of type EventType to other agents.
    #sends event EventType reference;

    // The agent has a capability of type CapabilityType.
    #has capability CapabilityType reference;

    // Data members (Java data structures). Constructor method.
    AgentType(arglist)
    {
        super("agent name");
        :
        :
    }
    // Java methods that implement agent functionality.
    // (These may be called from within the agent's plans.)
}
```

Figure 5.15: Code Template for Developing a Simple Agent in JACK

Furthermore the JACK allows drags and drop function (see. Fig 5.16) for software development which makes agent development as intuitive as developing objects in Object Oriented (OO) programming.

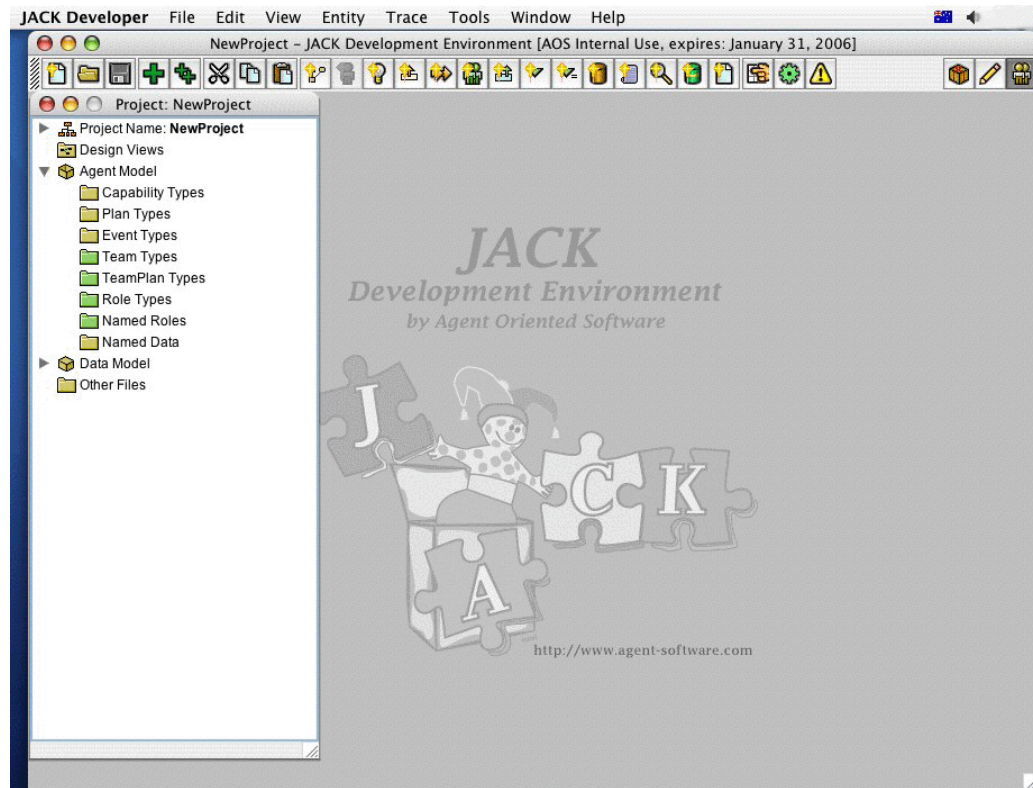


Figure 5.16: JACK Programming Environment

Thus, JACK has a good theoretical framework for agent development. It expresses that the agent is coded and, when the code is compiled:

- an agent gets instantiated (deployed into its environment);
- an agent waits for an event (that provides a goal satisfaction requirement) that it is supposed to respond to; or
 - if it receives an event to initiate a re/action that can handle this event/goal;
 - or
 - if the action so initiated does not satisfy the goal, then it has to try a different plan.

This structure can be supported by the proposed 'agent diagram' described in Section 5.3.1 for mapping implementation (see Figure 5.3). The code template for developing a simple agent in JACK is shown in Figure 5.15.

The main reason of using this structure is the fact that Tropos is fully adept in designing agent systems, but before actually implementing them we have to revert back to object-oriented designs, which could cause many agent concepts to be lost during coding.

The origins of producing a comprehensive programming environment for agents can be seen from the extensive work of Travers (1996), with many attempts to make agent development easy and robust. Travers' (1996) initiation into languages like Agar and BrainWorks clearly shows the complexity that is to come when developing an agent-oriented programming structure.

5.4 Data Sets for Testing an Agent Variogram Modeller

Finding the best fit model for spatial dependency (constructing and fitting a variogram) usually requires the knowledge of an expert to achieve it, as discussed in Chapter 4. In this thesis, a distributed modeller agent system aims to be built and tested in order to investigate distributed component GIS. This agent system will be termed here as variogram agent component (VAC).

A well known published data set in geostatistics is the Walker Lake data, which will be used to test and validate the implemented agent system VAC. The Walker Lake data had been previously analysed by Issaks and Srivastava (1989). Walker Lake is a perennial terminal lake located in the Great Basin of Western Nevada (near the California-Nevada border), in the western United States. Its volume was measured in 1994 and found to have a fluid volume of 2,060,000 acre-ft over an area of 50.3 square miles (130 km²). However, the data was not derived from the lake itself, but from its surrounding area. The data set used by Isaaks and Srivastava (1989) was derived from

a DEM of the National Cartographic Information Center (NCIC) from the lake area. The data was obtained from the DEM coded in the Defence Mapping Agency Digital format. It consists of an area covering 1° of latitude by 1° of longitude with a ground distance between adjacent points of approximately 200 feet. Thus each 14° by 1° quadrangle contained about 2.5 million elevation points, covering a standard 1:250,000 topographic map sheet over two contiguous blocks.

Three variables have been derived using this DEM, for 78,000 points on a 260 x 300 foot rectangular grid. This is known as the exhaustive dataset. This data grid contains an identification number for each point, the x, y location of measurement and the three values V, U and the indicator T. V and U represent soil contaminants in parts per million (ppm) and T is the time interval. The data is discontinuous and highly skewed. However, the version of the data used by Issaks and Srivastava (1989) is not exhaustive, as it only contains data from a selected region containing 470 points (known as the sample dataset). Only the data examined by them will be used here, as it is sufficient for the required test. In Issaks and Srivastava's (1989) publication, their data is presented on pages 115-119. The distribution of points of the Walker Lake sample data set, Nevada is given in Figure 5.16, while the file specification is given in Table 5.1.

Table 5.1: Explanation of the Walker Lake Data

Symbol	Explanation
Id	Identification Number
X	X location in metres
Y	Y location in metres
V	V variable, concentration in ppm
U	U variable, concentration in ppm

This data will be used in this thesis for testing and validation, aimed at determining the functionality and performance of the VAC. The outputs produced by the VAC will then be compared to those derived by the original authors, Issaks and Srivastava (1989). The

main reason for using this data is that it has many features which pose a range of challenges often encountered in geostatistics. These include outliers, clusters, trends and errors that need to be removed before any variogram modelling can be applied, if useable results are to be achieved. Issaks and Srivastava (1989) outline these challenges as:

1. the description of the important features of the data;
2. the estimation of an average value over a large area;
3. the estimation of an unknown value at a particular location;
4. the estimation of an average value over a small area;
5. the use of the available sampling to check the performance of an estimation methodology;
6. the use of sample values of one variable to improve the estimation of another variable;
7. the estimation of a distribution of values over a large area;
8. the estimation of a distribution of values over a small area;
9. the estimation of a distribution of block average; and
10. the assessment of the uncertainty of estimates.

In this thesis, the 470 points will be used as the test data set, where a number of sample blocks will be segmented from the full set. Each block will be independently tested and the results compared. As each block will be minimally defined, more data might be required during the experiment and so will be added accordingly. This is intended to build the basic structure of the experiment in order to test the VAC and see its performance. The test results will be compared to those obtained via just simply using conventional geostatistical tools.

Most of the technical experiments of geostatistical components (e.g. trends, variograms, kriging, etc.) are based on the V values of the Walker Lake dataset. A map showing the data posting of the V values is given in Figure 5.17.

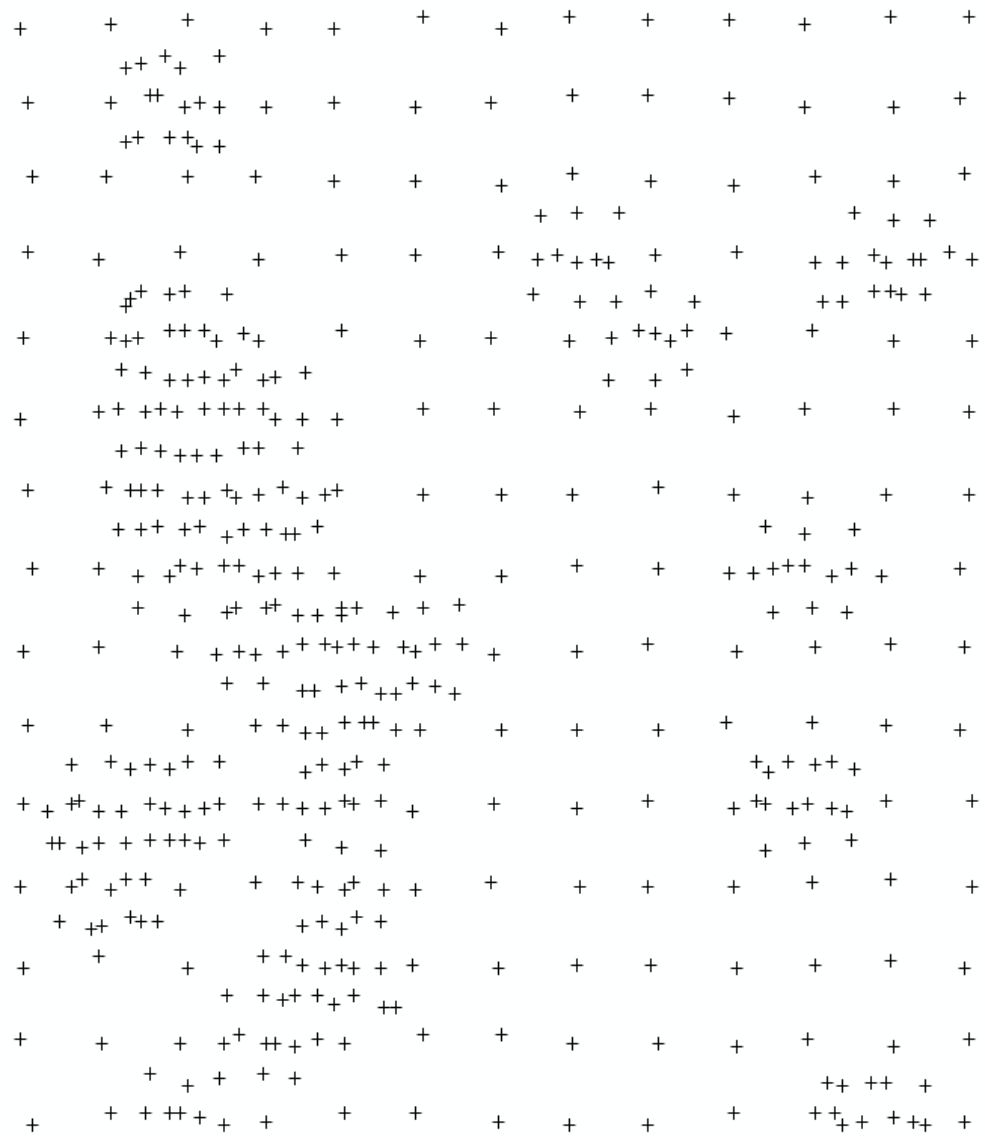


Figure 5.17: Concentration and Clustering of V Values in the Walker Lake Dataset

CHAPTER SIX: THE DESIGN OF THE VARIOGRAM AGENT COMPONENT(S)

6.1 Introduction

This chapter looks into the modelling (analysis and design) and the implementation (frameworks, platforms and coding algorithms) of the agent proposed in this thesis. Designing an agent based system is challenging for various reasons, mainly due to complexity of the agent development platforms and formalise Modelling language. The available modelling languages will be assessed for distributed component GIS with consideration for their suitability to support implementation on JACK programming environment.

The agency features will be examine and assessed for their ease of implementation and function with the chosen modelling and implementation platform. New agency features will be studied and embedded in the software development process. This chapter will define the functionality of individual agent while determining their collective and collaborative performance.

6.2 GIS Agent-based System Architecture

Here a framework is defined as a supporting structure in which a general platform for agent supporting component based GIS service can be produced. It is set to be the underlying infrastructure and provision for any distributed agent component GIS. The platform to be structured will provide the underlying infrastructure to hold different components of GIS. Thus, using this framework and platform, another VAC project can be analysed, designed and developed. The overall architecture of the agent system for GIS is based on a three-tier structure (see Figure 6.1), which support one another. The top level tier has to conform to the requirements of the lower level. These tiers are:

1. The framework, which provides a set of protocols (policies and regulations) for a chosen platform to conform to. The framework defines the communication mechanism (i.e. TCP/IP); the core structure of the agents (i.e. what should individual agent provide to other agent and acquire, and the data type and format); the function of the agent (i.e. the environment which an agent can act upon, extent it can communicate, and capability).
2. The platform, which is the underlying engine upon which a component can be developed. The platform consist of programming engines (i.e. JACK, Egglets, JADE, MASE and more advance O-MASE), language (i.e. C and Java) and agent code design structure since the current available engines are not fully agent oriented but Object Oriented they are not fully sufficient for developing agent and thus new structures need to be developed).
3. The actual component that performs a specific function. This component is an agent with its own goal. Furthermore, it encapsulates sub-agents with individual sub-goals that help achieve the main goal. In this thesis the sample is the Variogram agent but could be remote sensing agent, cartographic agent, survey agent, and/or any other geographical components.

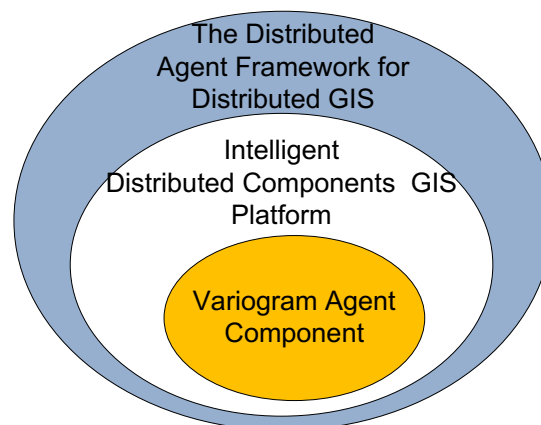


Figure 6.1: Architecture Design for Component base GIS System, Showing Three Tier Structure

6.2.1 The Distributed Agent Framework for GIS Agents

The main aim of the framework (shown in Figure 6.2) is to provide the basic design protocols and infrastructure in which agent-based GIS components can be deployed. To achieve this, it is important to define the primary requirements to develop GIS agents. According to the literature analyses (Sánchez et al., 2005), these are:

- knowledge acquisition - the learning ability of the agent. In the proposed framework/tool/platform, it is envisaged that an agent will learn by interacting and communicating with GIS experts (humans) and other software systems (including other agent components);
- knowledge verification – the possibility of determining the status of its environment and acting upon it. In the proposed framework, the agent will sense its environment when a certain event takes place and then will act upon it with the knowledge it has at that point. If the agent does not have the appropriate knowledge (usually in the form of plans) to act, then it will try to consult a human GIS expert or other agent(s), through its interfaces, to obtain the required knowledge. Then, the newly acquired knowledge will be added as a new plan during the run time of the agent system; and
- knowledge dissemination - the ability of understanding its own capability and then disseminating it to other agents that might need that information. Since an agent has to share its knowledge, this makes for a cleaner structure by providing knowledge (plans) without needing to amend the internal structure of the agent being taught.

This structure is based on a continuous flow of requests and proposals from communicating agents. Immediately after agent creation, it will assess the resources (processor, memory, communication and storage) of its situated physical environment in order to determine its capabilities. Similarly, Batty (2005c) explained the importance of framework while compare various computational spatial models to determine a general framework for Geosimulations using agents. In this thesis the main goal will be to obtain knowledge available in its domain and enhanced it by observing other agents,

human expert and, particularly, the pattern of the expert's decisions. The framework described in this section effectively defines an agent-based GIS technology, where a base platform is provided that allows the deployment of agents of distributed component GIS. In particular, the proposed framework indicates the ability of each agent's (that belongs to an agent-based system following the framework) communication protocols and its responsibility to the system. Each agent is able to acquire knowledge dynamically and use this knowledge to teach other agents.

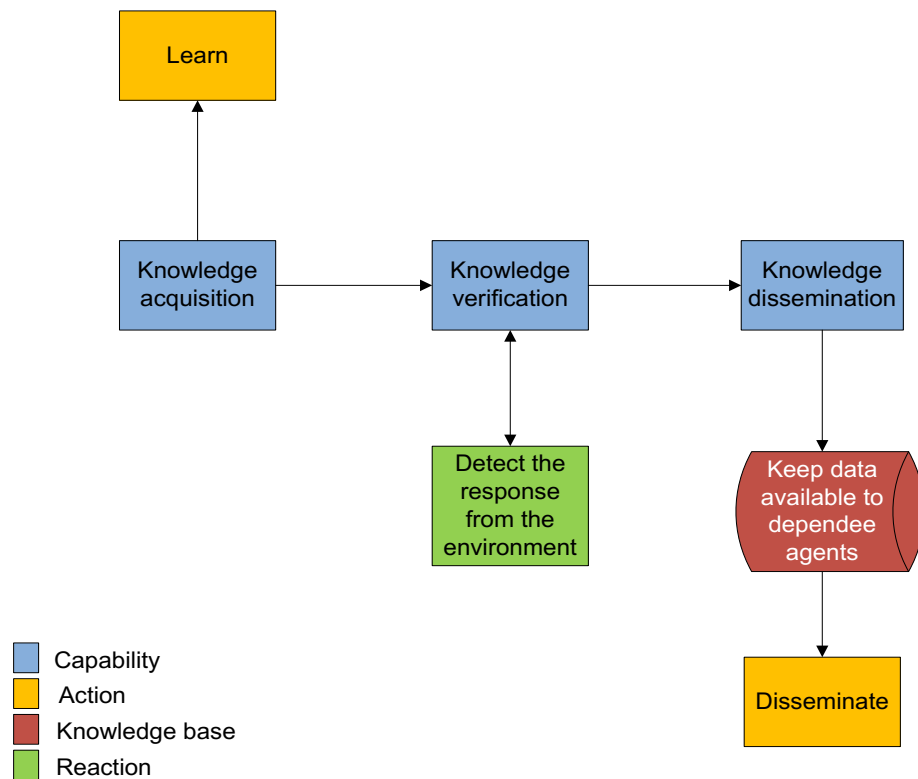


Figure 6.2: Fundamental protocol which all agent components must adhere to within Distributed Component GIS System

6.2.2 The Platform for Intelligent Distributed Component GIS

The architecture is implemented using existing agent engines. The chosen engine for the implementation of the experimental prototype is JACK Agent which is developed over a Java Virtual Machine. This provides the benefit of portability flexibility since Java is platform independent (can run over any Operating System, i.e. Windows, Mac, UNIX,

etc.). Also due to Java being an open source environment there is much availability of application programming interfaces (API), e.g. Graph drawing interface, Math and Statistics APIs were used for this project. Since Java was used as underlying infrastructure for JACK (by Jack developers) it was used to the advantage of the developing of the VAC. Over the JACK architecture a specialised platform was developed. The platform has specialised protocols and structure (fig 6.2) to allow a common ground for agent runtime binding, communication, collaboration and the ability to identify each other's existence, role and functionality. On this platform VAC was developed as a prototype component to the Distributed Component GIS System. In this thesis this component is design as Multi-Agent System with features explained in section 6.4.

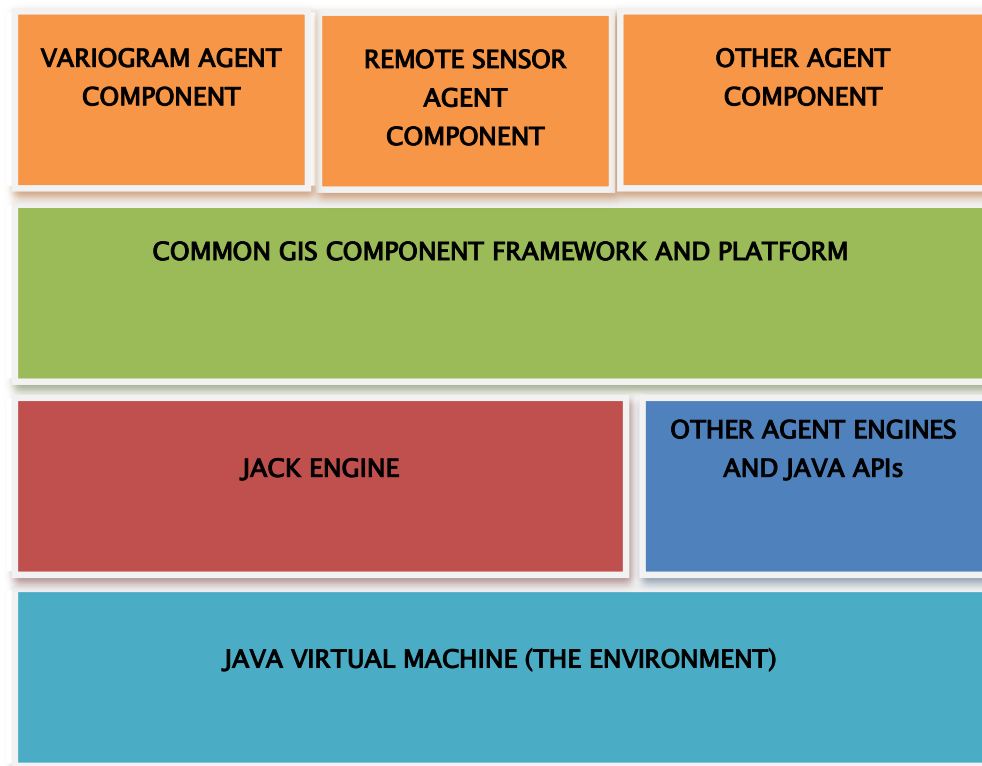


Figure 6.3: General Architectural Platform for and Agent-based GIS

Thus, the overall architecture comprises of a framework which defines the rules and requirements of the agent systems which would facilitate the overall aims of the

distributed components GIS, with the ability for self-growth and greater flexibility. On top of this framework, a general platform (see Figure 6.3) is embedded. This platform promotes inter-agent communication, which is an important factor. More importantly, it provides an enabling environment for system developers and geographers, and acts as a reference architecture for developing distributed agents, as long as the agent being developed sits on the common GIS agent platform. Thus, this platform represents the physical development environment, where all the agents adhering to the given framework (despite being developed separately and irrespective of their function) should be able to work together through the platform to achieve certain goals.

Each agent is expected to work autonomously to achieve its goal. These individual agents could be distributed remotely to one another. The distributed agents could be facilitated by their own multi-agent systems and provide a semi or fully functional GIS component. Here, one of these components is developed which represents a geostatistical tool, the VAC.

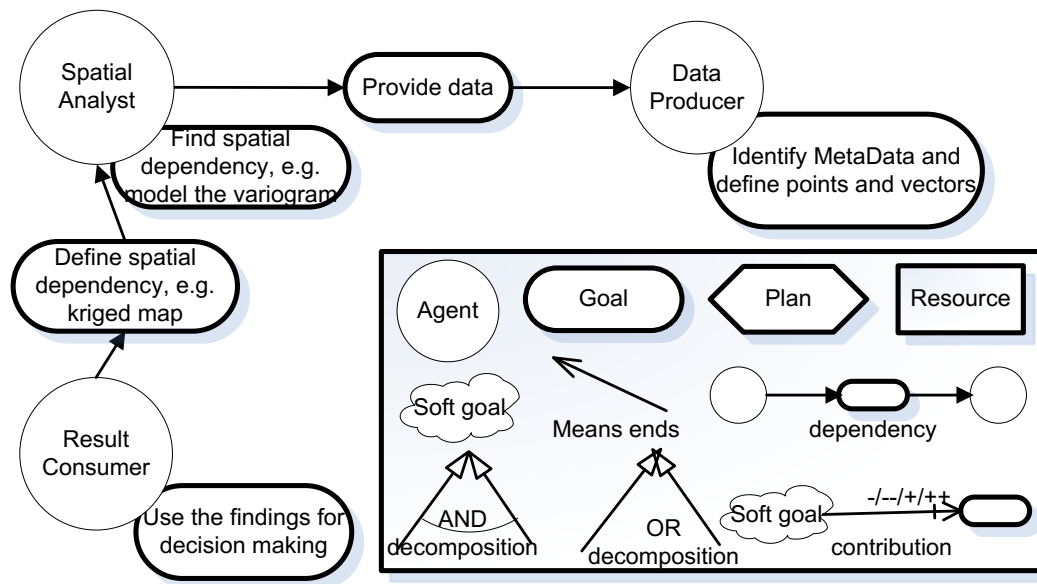
The development of the VAC is intended to show the functionality and limitations of the overall architecture to the suggested technology. In terms of the tool being developed in this thesis, the problem we are concerned with is the ability of the variogram agents to be able to do what they are supposed to do in terms of representing agent characteristics and, to an extent, the problem-solving and learning mechanism.

6.3 VAC Design Using Tropos

The agent system architecture has been designed and developed using the Tropos methodology, which not only allows analysis of the system, but also supports the analysis of its environment and how the system environment (and all the related stakeholders) might affect the design of the system. In Tropos, the system and its environment is modelled in terms of actors which have strategic goals. They might have a number of dependencies with other actors for the satisfaction of goals which the actors

cannot achieve on their own. Using the Walker Lake sample dataset described in Chapter Five for the VAC, three main actors (stakeholders) can be identified, as shown in Figure 6.4.

1. The **Data Producer** actor. This is the actor responsible for the production of the data (e.g. the Walker Lake dataset). In a typical GIS scenario, the input actor is usually a human, who imports the data from a data producer (provider).
2. The **Spatial Analyst** actor is the person who wishes to process the dataset in order to address the types of problems listed in section 5.4 (from Isaak and Srivastava, 1989).
3. The **Result Consumer** actor is responsible for decision-making based on the outputs of the VAC.



Note: Here we display a legend that applies to all Tropos diagrams used in this section.

Figure 6.4: Actor Diagram for the VAC System

Following the Tropos process, each of these actors is further defined in terms of his/her goals and tasks. Various alternatives are analysed in order to select the best possible way of satisfying the actors' goals (Yu *et al.* 2007). This analysis allows us to reason

(amongst other things) whether an electronic system is needed and the advantages it provides over a manual system. The system analysis (shown in Figure 6.4) thus indicates the need to provide a software system to support the variogram definition and satisfy the main goal of the Spatial Analyst actor to determine spatial dependency in data through the means of the variogram faster and more efficiently. The goal diagram of the Spatial Analyst actor is shown in Figure 6.5.

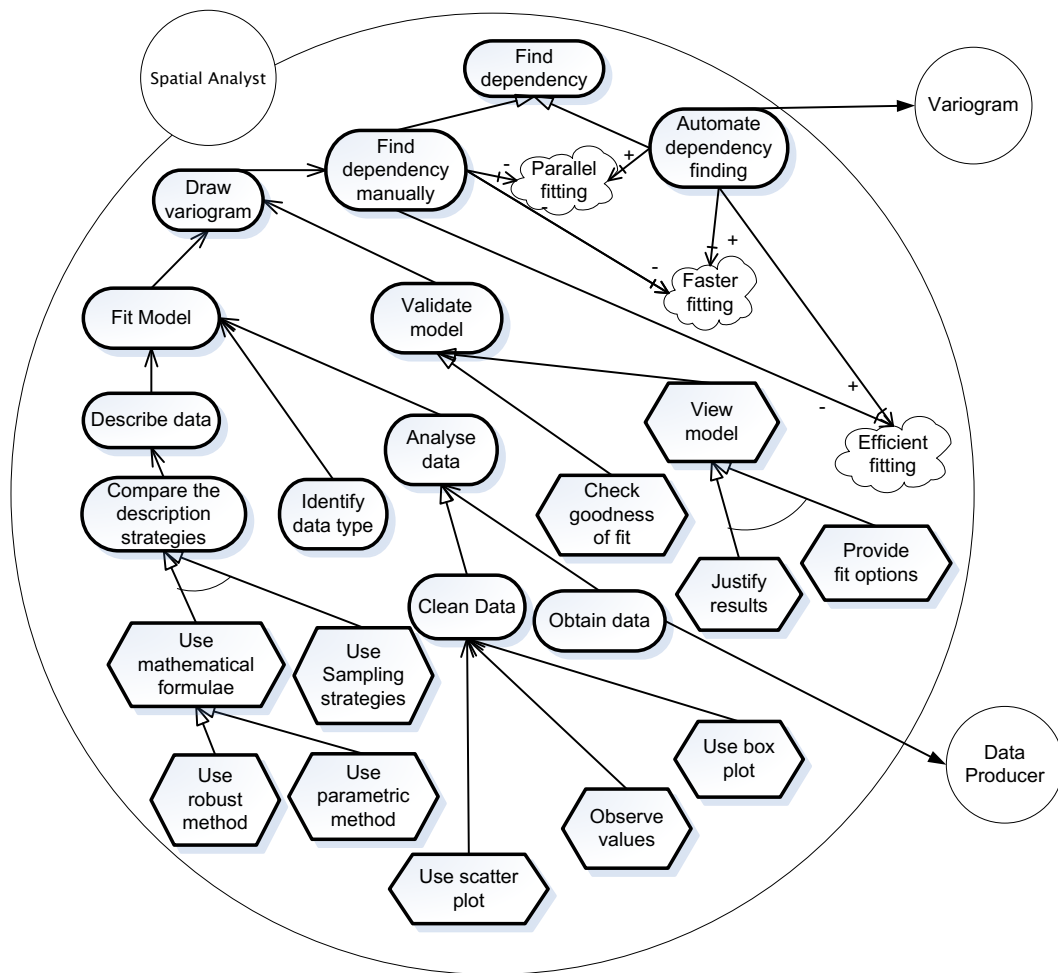


Figure 6.5: Goal Diagram of the Spatial Analyst Actor

As shown in the internal analysis of the Spatial Analyst actor, whose main goals can be achieved either electronically or manually. However, the issue of performing a faster,

parallel and efficient fitting is evident. The output of the above stage is the input on the late requirements stage of the Tropos methodology. In that stage, the proposed electronic system is defined as another actor that interacts with the actors shown in Figure 6.4. Using the same reasoning techniques utilised in the previous Tropos stage, the system actor is analysed in terms of its goals and tasks. For this reason, a variogram actor is defined. The system analysis of this actor is shown in Figure 6.6.

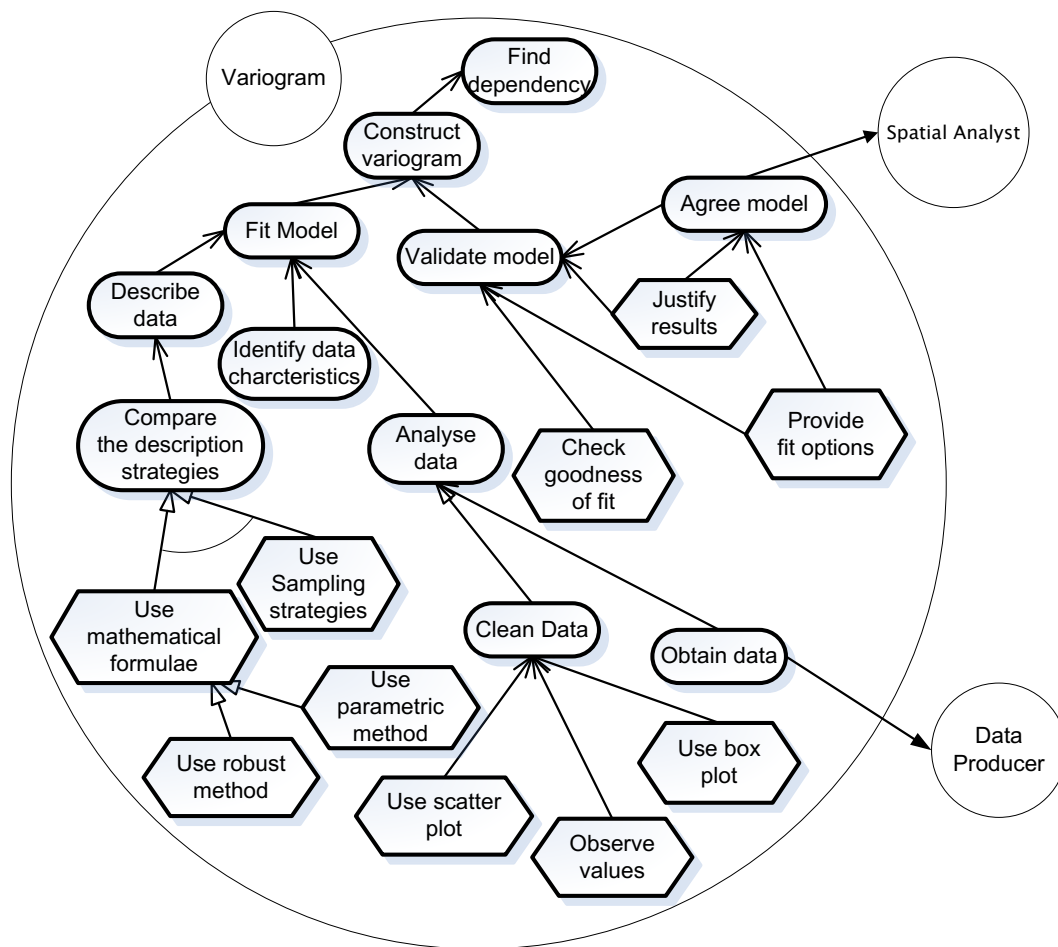
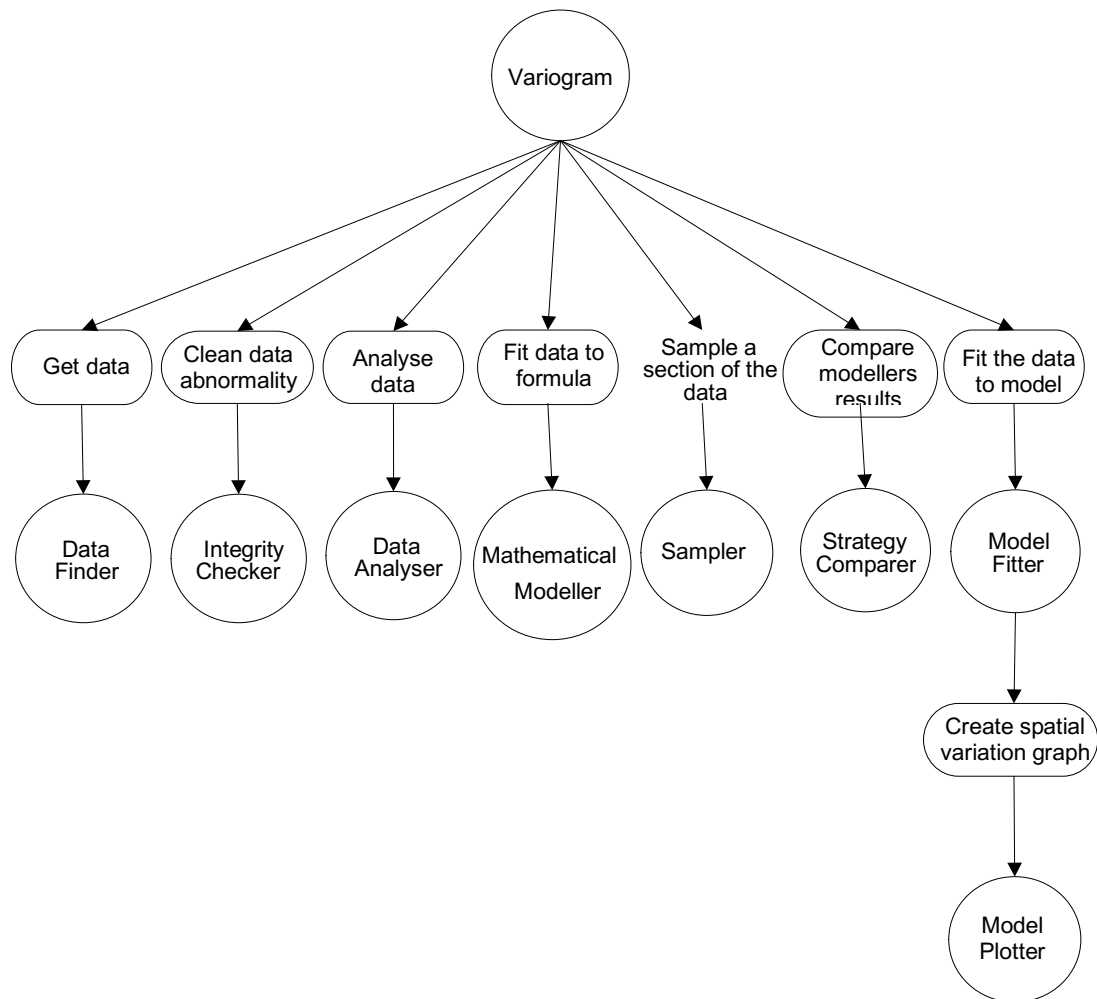


Figure 6.6: Goal Diagram of the Variogram Actor

The Variogram actor is the main actor in the GIS agents system and is the VAC, and would be just one component available within distributed components GIS. The Variogram agent has its own capabilities and goals, which are decomposed into sub-systems. Thus, when all the goals and tasks have been identified by the system's Variogram actor, the next stage of the methodology aims to decompose the system into

a set of agents each responsible for satisfying the system's goals and tasks. These capabilities include communicating with other actors, allowing its internal (sub-) actors to cooperate and share their knowledge with other GIS systems. The actor decomposition of this sub-system is given in Figure 6.7.



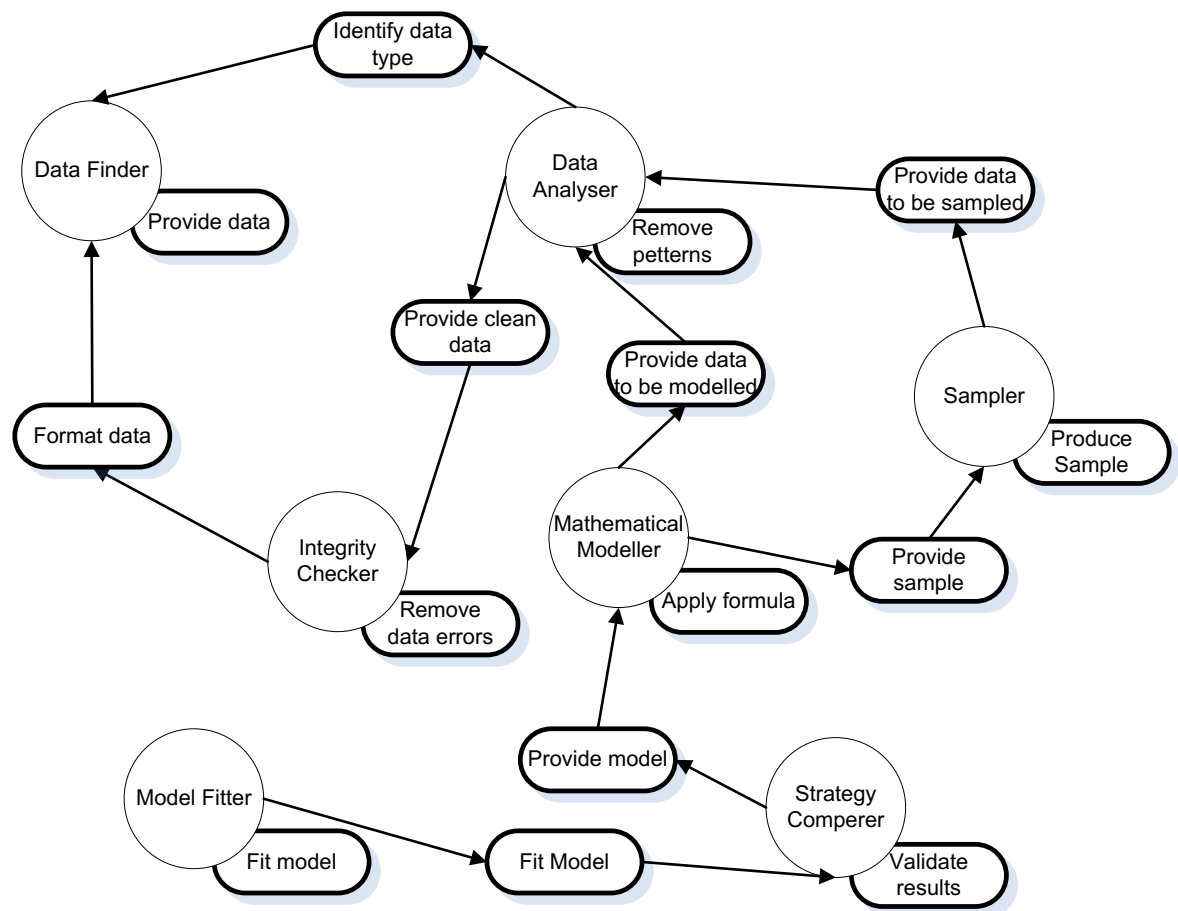
Note: Actor decomposition of the variogram agents - these actors are going to be defined as the agents in the system.

Figure 6.7: Actor Decomposition of the Variogram Agents

The proposed system has been analysed and designed with the aid of the Tropos agent-oriented software. The agents themselves have been constructed using JACK agents. The diagrammatic programming structure of the Variogram agent and all its sub-agents can be found in Appendix C. The main actor is the Spatial Analyst, requiring construction and modelling of a variogram. The human actor Spatial Analyst is converted to a

software agent called the Variogram agent. The Variogram agent acts as the container and regulator of its internal actors, which will be defined as the agents of the system. This concept is referred to as agent decomposition. Decomposed into a set of agents, each is responsible for satisfying the system's goals and task results in the following agents (see also Figure 6.7):

- **Data Finder agent:** responsible for acquiring data, establishing certain characteristics of structure and type, then providing it to the rest of the system;
- **Integrity Checker agent:** responsible for determining errors and cleaning the data (from problems like abnormality and sort). This agent performs box plots and scatter plots on the data to determine outliers and possible erroneous data;
- **Data Analyser agent:** responsible for determining if the type of data (or similar) has been received before or is a new data type, and so determine its structure. If the data has trends and/or clusters, it removes them and registers it as a new type of data;
- **Sampler agent:** responsible for determining sample sizes for the modelling and integrity checking of the variogram. This is done by defining a size of a section of the data that can give the most accurate spatial pattern prediction and hold back enough data to compare the results of the variogram;
- **Mathematical Modeller agent:** responsible for applying the appropriate formula for the calculation of γ given a particular data structure;
- **Strategy Comparer agent:** responsible for checking the results produced by the Sampler agent and Mathematical Modeller agent, and determining the residual errors; and
- **Model Fitter agent:** responsible for checking the plots and providing a best-fit curve as a model. This is done by using the plots defined by the Mathematical Modeller and/or Sampler agents and fitting them onto one of the validated curves (spherical, Gaussian, pure nugget, linear etc.). The actual drawings are achieved using a sub-agent, the Plotter:
 - The **Model Plotter agent** is responsible for plotting the various models chosen by the Model Fitter agent.



**Figure 6.8: Variogram Interagent Depender-Dependum-Dependee Diagram:
Internal Variogram Actors and their Dependencies**

These actors are defined as the agents of the variogram system. The information about these actors (agents), their goals and plans will be presented in order of their function and dependency. Figure 6.8 shows their depender-dependum-dependee diagram. Each of these agents will be further analysed and defined in more detail. However, apart from having these sub-agents to aid it to do its job, the Variogram agent has its own capabilities and goals. These capabilities are:

- communicating with other agents; and
- allowing its internal (sub-) agents to cooperate and share their knowledge with other variogram agents.

These functions set the minimum requirement for GIS agents. The actual information and action flow of the Variogram agent in relation to these agents is shown in the agent action diagram (see Figure 6.9):

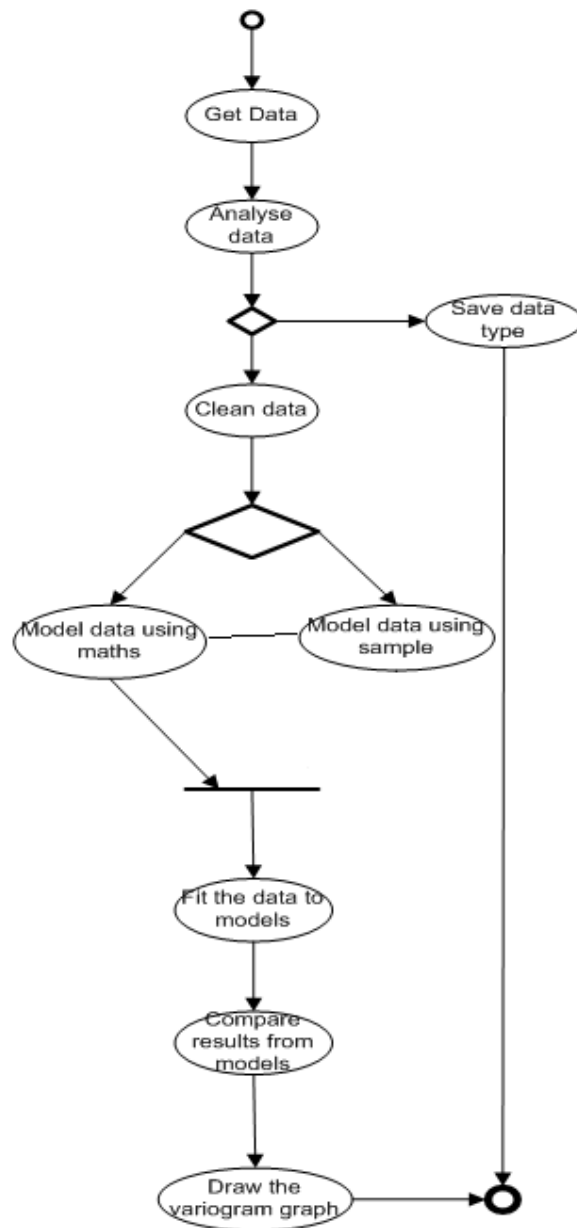


Figure 6.9: Algorithm for the Variogram Agent

6.3.1 Data Finder

The first agent in this architecture and simplest in term of performance is the **Data Finder**. The Data Finder agent is versatile and able to determine any new data that needs to analysed, whether from single or multiple sources. The source could be a human input, remote sensing portals, other computing devices or any other data distributing devices. The agents can also try to traverse the network for any information (data) that could aid the scenario. This agent's goal analysis is given in Figure 6.10.

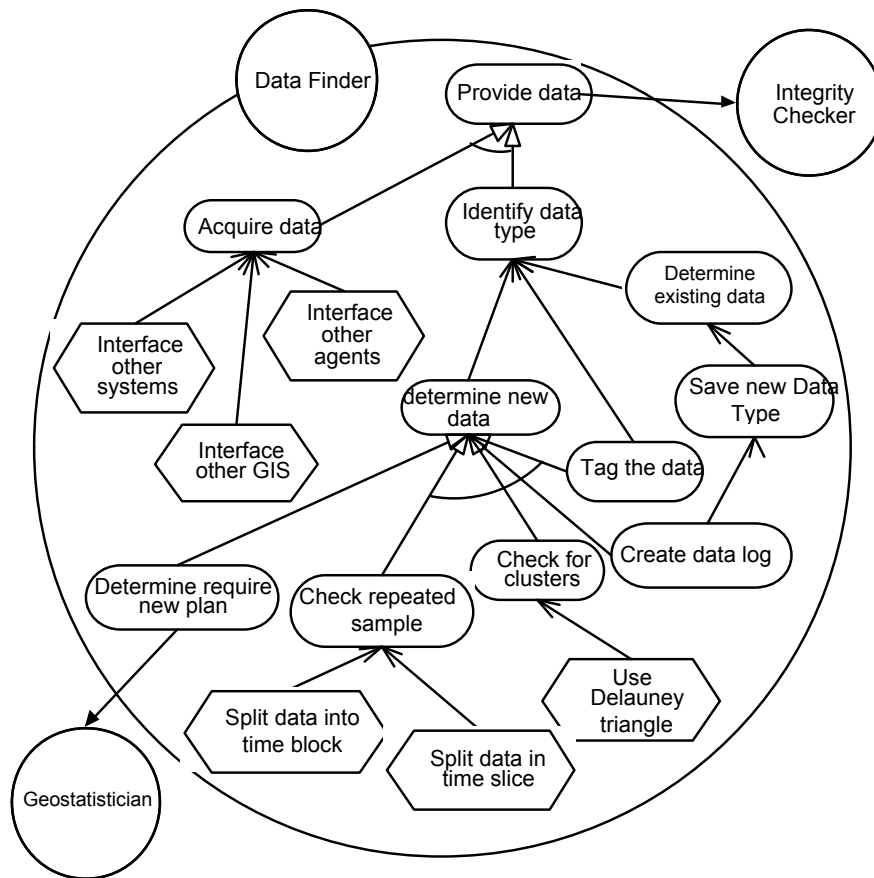


Figure 6.10: Goal Analysis for the Data Finder Agent

The Data Finder agent communicates only with the Data Analyser and Integrity Checker in the system. However, in the overall architecture it also communicates with external entities (could be agents, objects etc.) to acquire data. Its capabilities are to:

- acquire data, which is facilitated by two input capabilities that allow it to actually capture this data. The interfaces are:
 - direct data input: this function allows forceful data input into the agent. The data can be fed into the agent by a human agent or other GIS devices.
 - spatial data input: this data importing mechanism is normally initiated by the agent itself. This can be defined for specific time intervals.
- identify the data type, which checks if the data has:
 - clusters, and if the clusters have any significance to the data description.
 - temporal data, and what type of temporal data it is.

This agent possesses a plan which enables it to determine what would be required to produce a new plan, which thus displays the essential concept of dynamic binding. The requirement of dynamic binding function is explained by Krutisch *et. al.* (2003) albeit without the use of agent technology. When all the given plans have failed and using all the knowledge it possess, the agent will declare to the expert that it requires new plan and with the dynamic binding feature it can use this plan without having to restart the system. Furthermore, this new plan will be added into the agent's plan, so it can be used next time. To increase the functionality the reasoning algorithms will be embedded into the plans. The dynamic data binding mechanism is defined in detail in section 6.4.3 of this chapter.

6.3.2 Integrity Checker

The Integrity Checker is responsible for cleaning the data of any errors and determining if there is data that appears like an error but is not an error. It also determines if it has data that forms a cluster when looked at in the perspective of its neighbours. Figure 6.11 shows the Integrity Checker agent.

After realising the data structure and pattern given by the Data Analyser agent, the Integrity Checker will determine the correct formula to use. However, if the available

The main job of this agent is to clean the data and remove ambiguity. It communicates with the Data Finder and Data Analyser so that it can get the available data, clean the data and make it ready for the Sampler and/or Mathematical modellers to do their job.

6.2.3 Data Analyser

This agent is responsible for data analysis and determining if this data type is familiar. This is determined by checking the various properties of the data to be analysed. This is achieved by examining the scenario under study, which can be determined by:

- the spatial pattern (associated to the surface formation of the event or process), described in section 4.4.2 and section 4.4.3;
- the point event pattern (associated to the actual datum on the location x,y and its effect), described in section 4.4.2 and section 4.4.3;
- the subject under study (i.e. fluid, moving object, static object); and
- the behaviour of the subject under study (predictable or unpredictable). This can be measured using information like “if predictable, how predictable?”

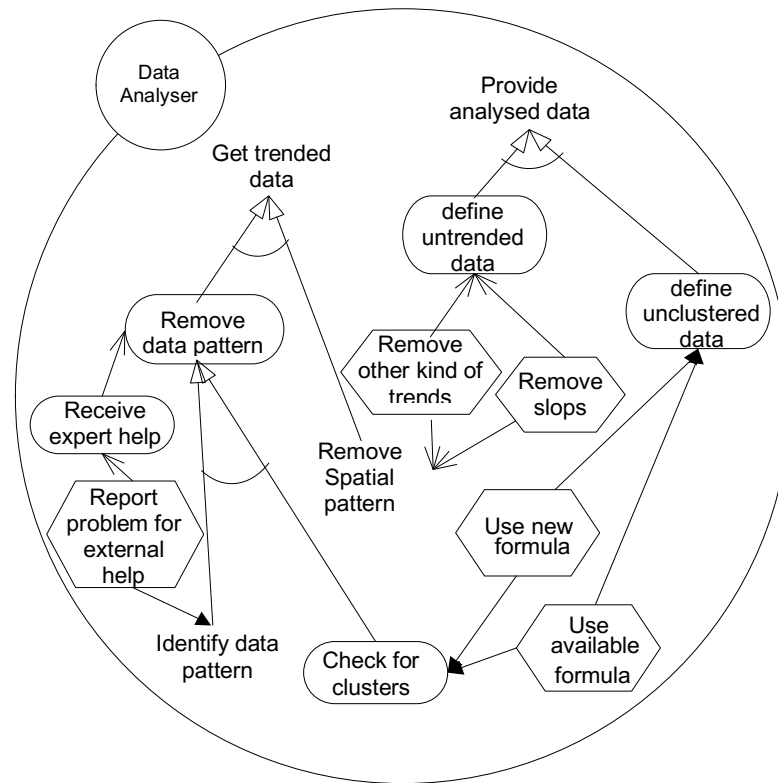


Figure 6.12: Data Analyser Agent Diagram Using Tropos Methodology

This agent is also responsible for removing spatial trends in the data, which will simplify the variogram creation for the scenario under study. Trend removal will be achieved using:

- polynomial regression;
- tessellation and triangulation; or
- general regression.

The choice between these techniques will be determined by the data type, which has already been established during the first goal achievement of this agent.

6.3.4 Mathematical Modeller

This agent is responsible for picking the right mathematical formula according to the data structure and pattern. So, its communication needs to be established in two phases:

1. It needs to communicate with the Integrity Checker (as the main communication agent). This will be the first strategy.
2. If the data is identified as being familiar by the Data Analyser, then there is the possibility of the Integrity Checker being skipped and the data being sent directly for mathematical processing. An example of this is where the Data Analyser agent is repeatedly processing the same kind of data in its normal duty. Thus, it will only send unfamiliar data, of which it has no previous experience, to the Integrity Checker agent.

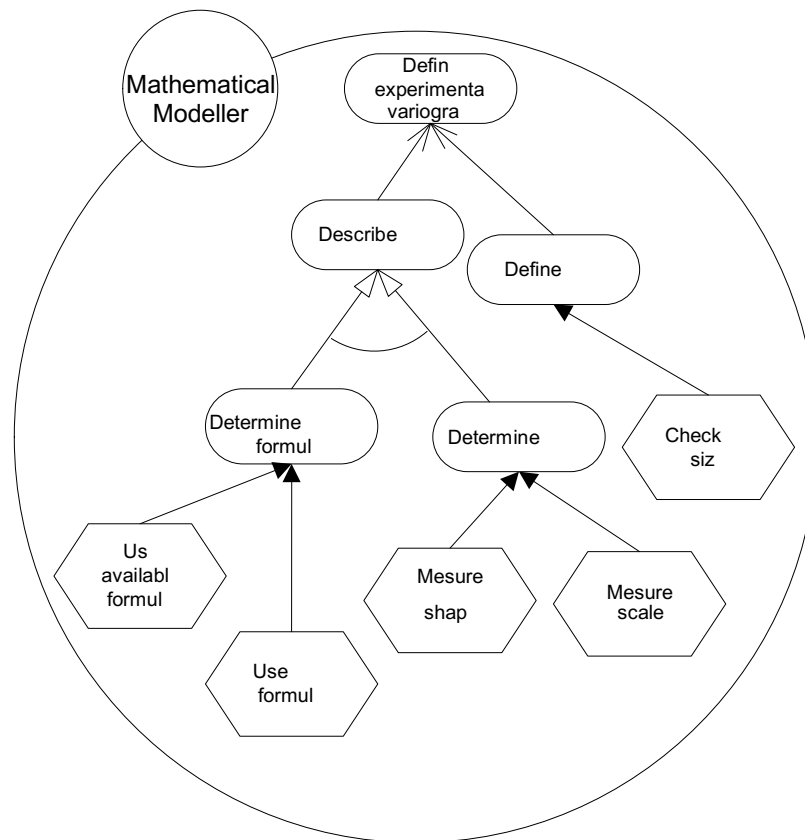


Figure 6.13: Mathematical Modeller Agent Diagram Using Tropos Methodology

This agent will have knowledge of equations with their preferred situation usage. Also, it will also have knowledge of defining a new formula on occasions where no available formulae are good enough for the type of data or situation. This new formula is then saved, to be used again if needed.

When a new formula is defined it is logged for verification, which is achieved through the reuse of the formula. Thus, every time it is used, its performance is detailed so that the agent or human expert can review it.

6.3.5 Sampler

This agent is called upon only if the dataset is large enough. From the data (which at this point all the properties are known, from the Data Analyser agent) it constructs different sampling schemes. This concept works like a game theory, where each sampling scheme competes with the rest to build the best possible variogram fit, as determined by the agent according to its current knowledge.

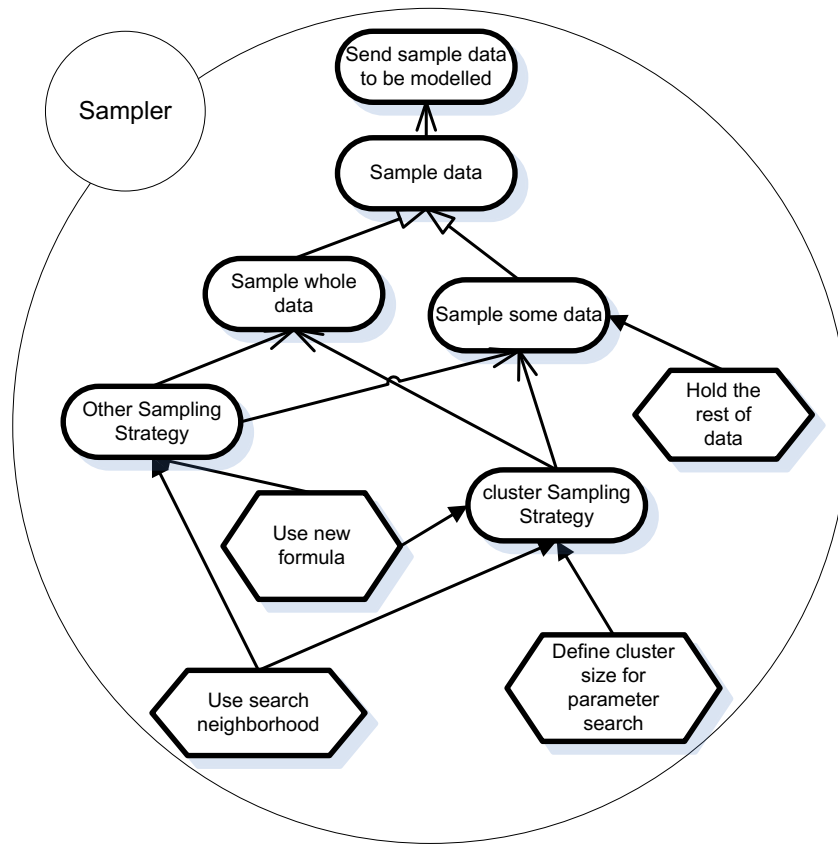


Figure 6.14: Sampler Agent Diagram Using Tropos Methodology

The result of each sample will be compared to the Mathematical Modeller agent and the closest result will be re-defined and tested to narrow the choice. This agent will hold information about the size, type, scenario and subject under study, as well as the data pattern, so the next time a similar sampling is required, it will minimise the time by simply using the closest matching sampling algorithm. This agent will follow the following action flow:

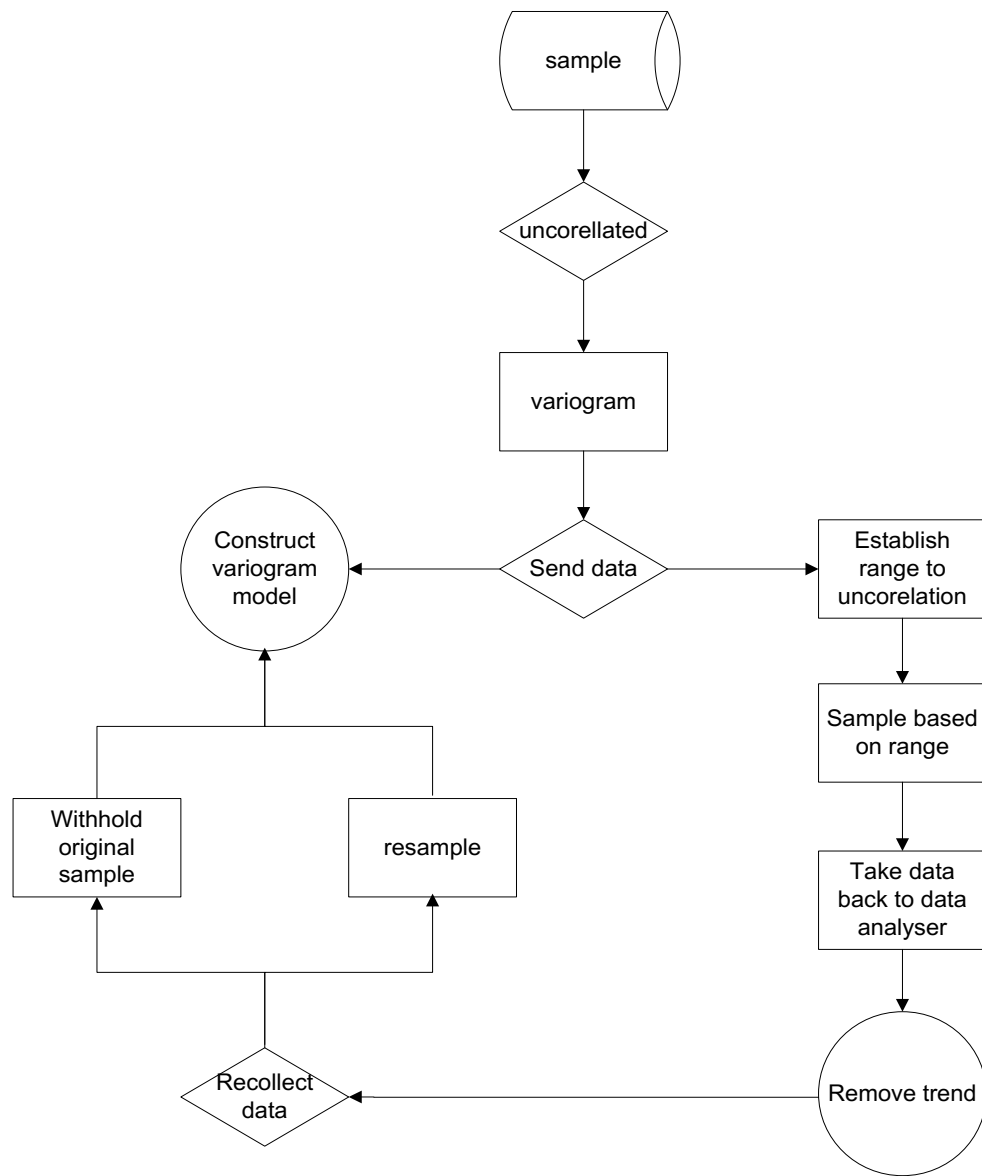


Figure 6.15: Sampling Action Flow Diagram

6.3.6 Strategy Comparer

The Strategy Comparer agent has only one plan. Its goal is to compare the sampling agent results and the Mathematical modeller results. The plan to is to compare the outcome and validate the sampling process. Thereafter it declares the data to have been processed Model Fitter agent to take over.

6.3.7 Model Fitter

The main function of this agent is to choose the appropriate model from the universal models available for the given data environment. Here, the universal models that are already known by the agent are:

- Spherical;
- Gaussian;
- Exponential;
- Pure nugget; and
- Linear.

Here the lag, range and sill are established for the parametric (mathematical) variogram and will be chosen by making sure that:

1. the model fit lies to the right of the mathematical model, otherwise change the scale (e.g. length value);
2. if more than 50% of the data lie to the right of the points estimated by the mathematical model and this lies on the same line as the model fitted, then it is good.
3. the range is the distance on the x axis to the plateau of the mathematical model.

A non-zero nugget indicates that repeated measurements at the same point will yield different values. The model chosen is also determined by looking into the following known indicators (Isaaks and Srivastava, 1989; Cressie, 1991):

- If the range is smaller than the sill, the spherical model is good.
- If the range is almost twice the sill, the Gaussian model is good.
- If the range is larger than the sill, the exponential model is good.

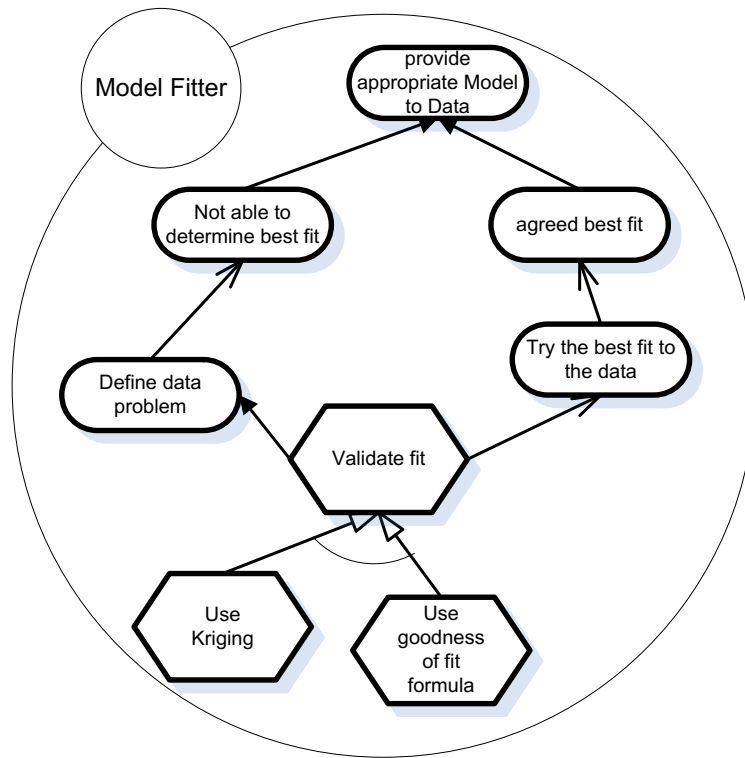


Figure 6.16: Model Fitter Agent Diagram Using Tropos Methodology

This agent is responsible for re-fitting the data to the given model if the Model Plotter returns that the strategy used to model the variogram is not satisfactory for any reason (according to the Spatial Analyst actor - i.e. based on a human decision). The plan then acts to have the data re-analysed and thereafter the data is tagged to what has been decided on its analysis and the reason as to why the original (first trial run) did not work, so that the technique that produced unusable results is not repeated in the future. This agent is also responsible for validating the goodness of fit. This is achieved using ordinary kriging and indicative goodness of fit to estimate a percentage of values and determine whether the variogram fit is good. This agent employs similar technique to that employed in Geovariences (2009) for automated variogram fit.

6.3.8 Model Plotter

This agent has the final task of displaying the model chosen for the variogram. It achieves this through the use of a specialised plan called the Plot Plan, which has the functionality to choose the appropriate representation scale for the GUI.

Even though this agent has the very simple task of simply representing the already chosen variogram model selected by the Model Fitter agent, it also has an important function to make sure the display is appropriate for the designated environment. An example of this could be that there are too many points on the graph and the screen is too small, which would cause the screen to be cumbersome and hard to read. This agent will point out this issue and attempt to either reduce the number of points or increase the sharpness of the viewable image. Also, in circumstances where it realises that there are issues in representing the current value, it flags to have the model refitted.

This agent is also responsible for communicating with the expert user, who can decide if the model fitted is inappropriate and ask the agent to refit. It has only one capability, called the 'model plotting cap', which allows this agent to achieve its goal. This agent is not included in the implementation of an experimental agent for this thesis. However, it is one of the important features and an important future requirement.

6.4 Algorithm, process and data handling for the VAC

At the start point, the data is searched within the approved locations. Currently, these locations are specified folders and the agents are triggered to work when the Data Finder agent believes that data has arrived into these bins. Then the actual data is scrutinised to find out if it has been used before. If the data is found to have been dealt with previously, it jumps all non-crucial steps that are normally taken to analyse the variogram, so that the agent does not waste time re-analysing it. Sometimes, it simply shows the previous result, where the expert user can say whether the data is to be re-

analysed and/or assist on a new strategy. Then, the Data Finder agent simultaneously checks for clusters and repetition. Heuristic methods will be used to check for repetition, including:

- checking the repetition pattern. If the repetition is perceived by the agent to be erratic and possibly due to an error, then it is removed; otherwise, it moves to the next strategy;
- checking for clusters, done using the formula Index of Cluster Size; and
- constructing the experimental variogram and finally fitting a variogram model to the data.

Determination of the slice is done heuristically, by checking if the data are divided into equal slices. This is only done if the data is found to have repetition. The main function here is to determine if the data is to be sliced for more analysis or not and, if yes, what structure should be followed. All other agents and their plans follow this same mechanism, where formulae are mainly extracted, as from Issaks and Srivastava (1989), Burrough and McDonell (1998) and Cressie (1993).

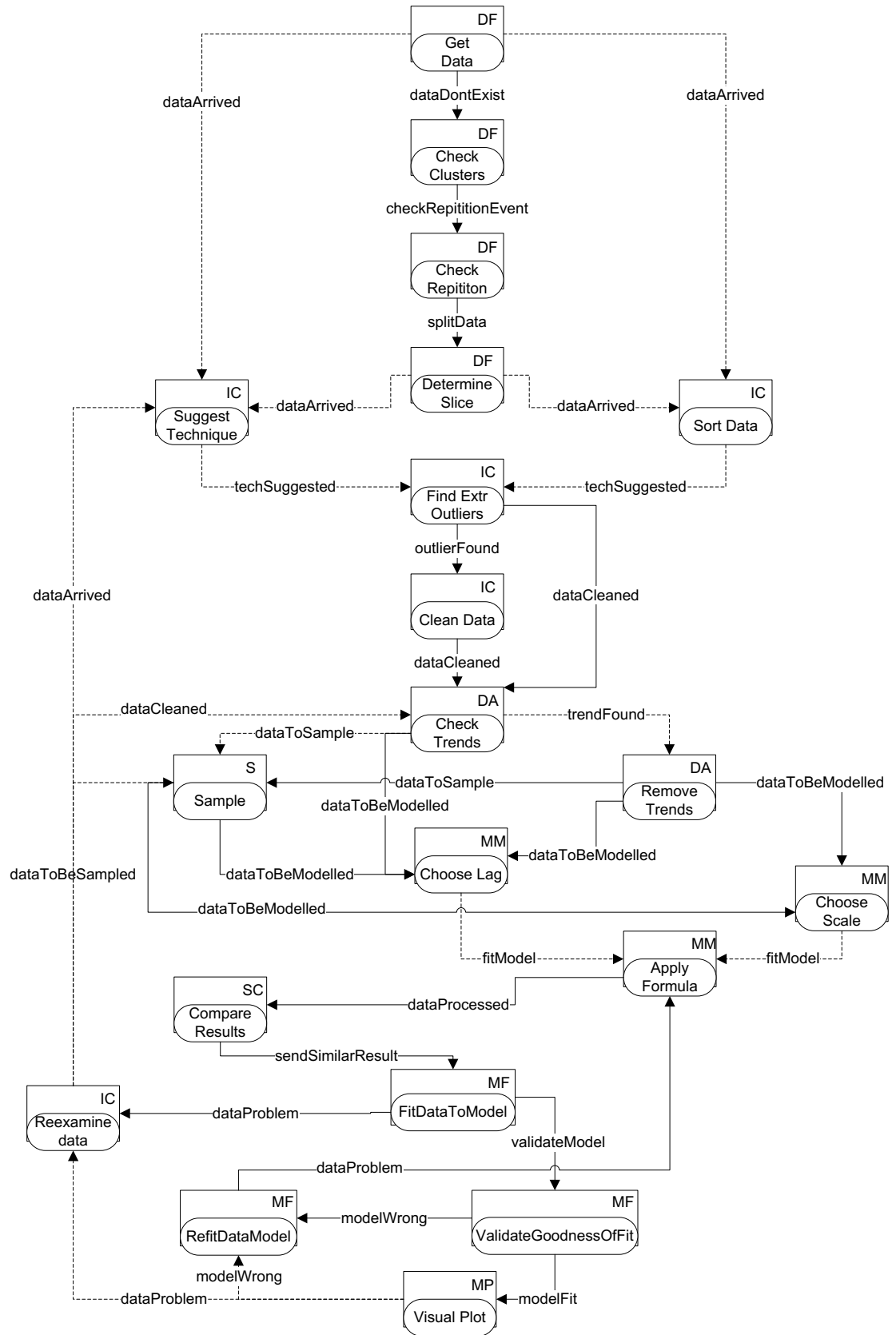


Figure 6.17: Agent's Plan and Event Flow

6.4.1 Data Handling of the VAC - Data Responsibility Diagram

The multi-agent system developed was able to handle the specific task of dealing with the geostatistical case at hand. Each agent was given the autonomous responsibility to do its job correctly and, by collaborating with one another, this can allow a form of social ability that gears them towards quick and decisive results. The main aim of each agent in the multi-agent system is specified by the data, resulting from its functionality. This is shown in Figure 6.18.

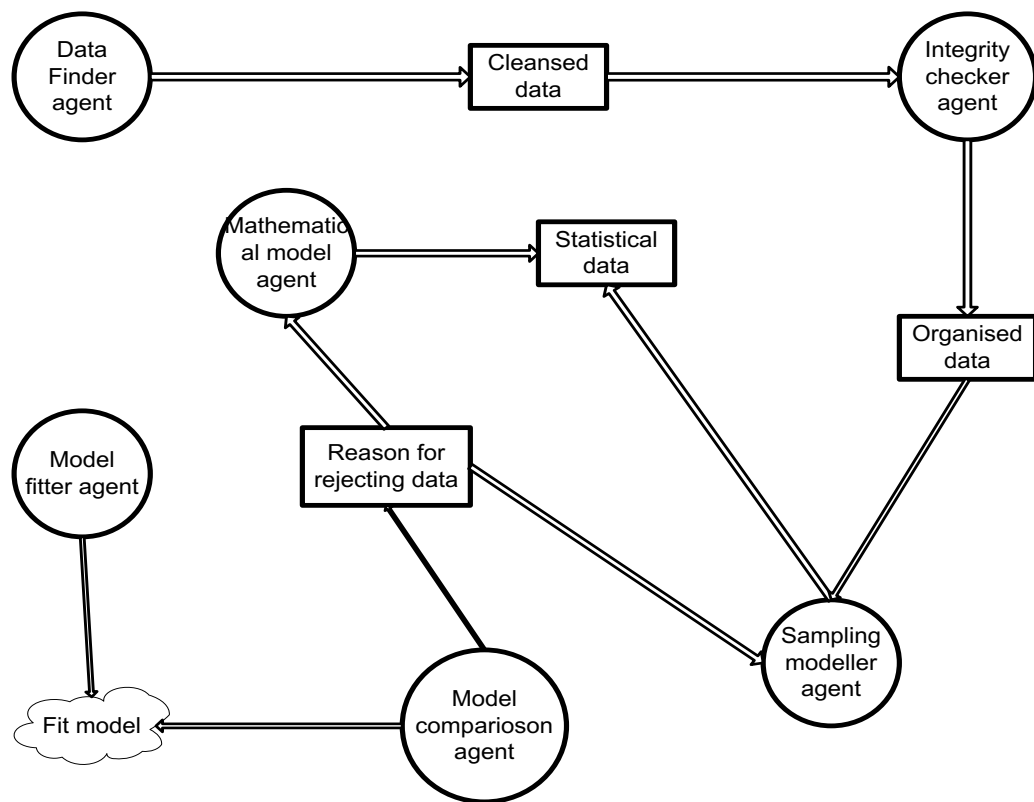


Figure 6.18: Data Flow Within Agents and Their Plans

Furthermore, these agents are able to multiply onto remote systems, in order to overcome any lack of capacity in local computing resources. This concept is defined in more detail later (see section 6.5). This concept is facilitated by the social ability of agents and networking mechanism provided by the agent programming platform. This feature was very hard to achieve using the JACK agent development tool, as this is

targeted towards providing BDI intelligence and easy internal communication (but not much external communication). Tools like Egglets and Voyager can easily provide this feature, as they provide an easy mechanism of code and data migration. In JACK, only data migration was achieved. Even this was only a kind of primitive migration, achieved by providing a predefined location where data is stored and, if the analysis is to be done by an external agent, the agent is pointed towards this location and can either copy the data to its designated (on the machine) location in its native environment or just do the analysis on the fly (the current location of the data). However, the latter will put the mobility or code migration on a very low level. The agent system for GIS requires mobility as one of its primary features, so to allow flexibility the agent was made flexible enough to be able to choose any one of these two mechanisms.

6.4.2. Agent Architecture: Design of VAC using Unified Agent Diagram

In software analysis and design, a structure providing the overall picture of the agent system was established in Chapter Five, section 5.3.1. Thus, here we prove the success of this new structure by showing the design of the VAC and using it in a real and well established spatial environment situation (the Walker Lake case, dataset in Appendix B). This is important, as it shows the whole system in its perspective environment, and also visualises agents to their functions and collaborators. For this, a diagrammatic interpretation of the overall agent system was determined. The unified agent diagram is used here to provide an outlook of the system. The agent architecture, as seen in the agent diagram, is defined in section 6.2. This architecture allows the agent to be visualised on a unified platform (see Figure 6.19).

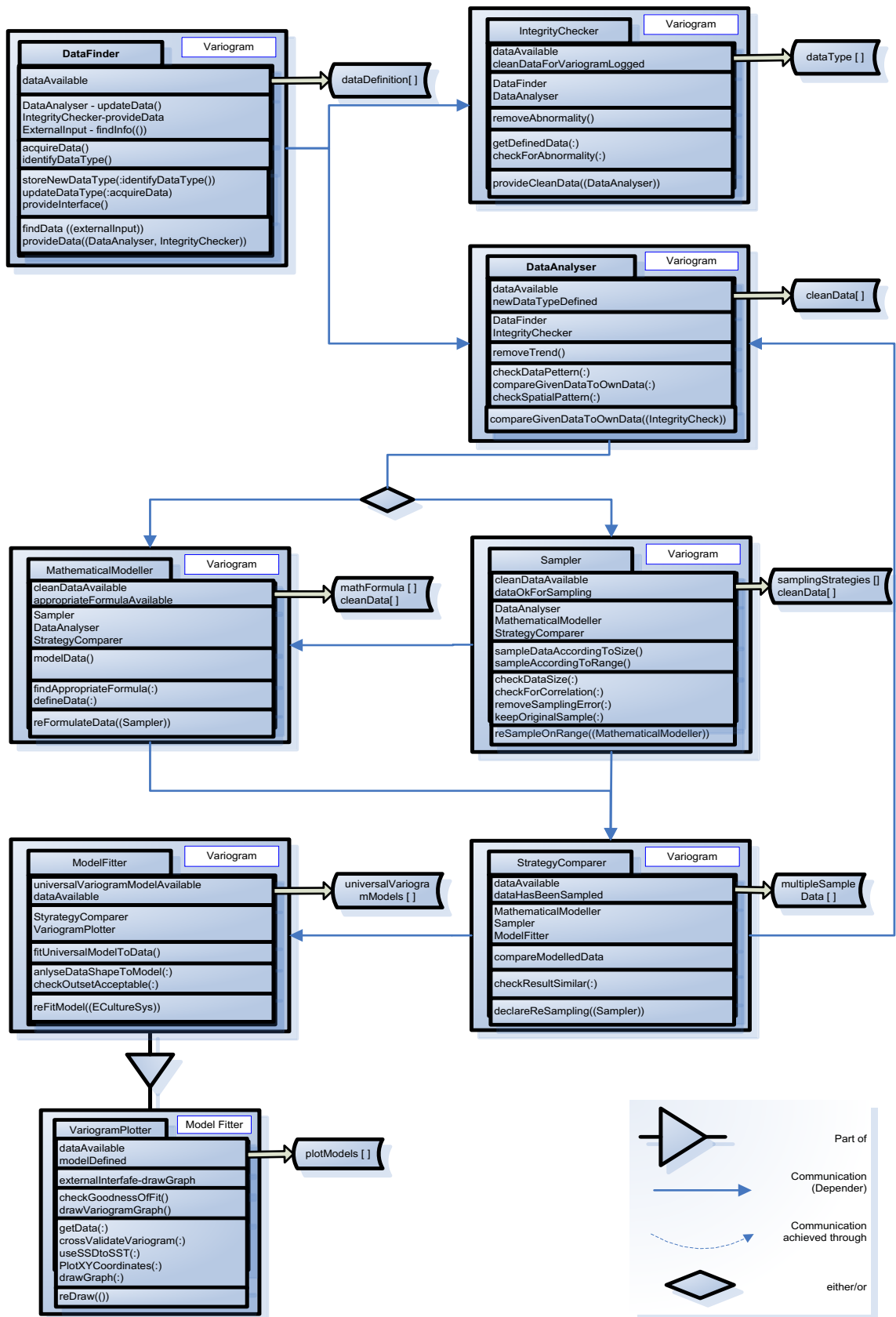


Figure 6.19: VAC design using unified Agent Diagram

Figure 6.19 provides an overall picture of the agent architecture and inter-agent relationship. This is intended to give a clear picture of the agent functionality to the GIS expert with not much software engineering knowledge. At the same time, this structure should allow an expert programmer to determine the mechanism and functionality of the overall system, by providing the actual programmatic notation of each agent, their function and how they achieve such function.

6.5 Required Features for Software Agents as Components for GIS

Developing an intelligent agent-based structure to communicate between GIS tools residing remotely from each other is an important concept in improving GIS technology. Furthermore, interoperation of different GIS platforms (developed using different binary structures (Operating systems and programming languages)) and collaboration of different tools can be achieved via this structure. Such collaboration will include agents teaching each other better techniques to tackle a problem. Also, as this structure will allow the GIS to be broken down into even smaller, more specialised, tools it will allow better modularity, functionality and execution speed, which are important factors in computational tools (computing in general) and in this case GIS.

The proposed technology is identified to especially benefit from specific characteristics of intelligent agents. The characteristics include reactive ability that will allow an interaction of GIS components to form a social behaviour in a similar environment and, most importantly, the autonomicity of individual agents. There are many features that characterise intelligent agents; however, for the VAC only some of the features were found to be important and considered, these are:

- autonomicity (described in Chapter Three) - probably the most important feature in terms of this research, as it describes the ability of agents to work explicitly towards their goal without depending on other agents (feature described by

Wooldridge (2003) which makes up a basic character of a software agent and used by all the agents in the VAC;

- reactivity, which expresses the need for an intelligent agent to be able to perceive its environment and respond to changes (internal and external) to the system, in order to consistently satisfy their design objectives or goals (Wooldridge, 2004);
- social ability, interacting with each other or external agents (humans, or a separate variogram in the case of this research) in order to satisfy their design objectives. This characteristic is particularly important when dealing with spatial scales of measurement, and specifically the utilisation of a multi-scale approach (Tate and Atkinson, 2000) in dealing with large or complex datasets;
- intelligence, use the BDI structure of JACK agents to decide the best solution to the situation. Furthermore, if the expert user determines the given solution to be unsuitable for the situation, to learn from this mistake and, in the future, use the new mechanism defined by the expert user during this incident to handle the data. This function is termed the 'dynamic binding mechanism' and was developed for the system used in this thesis, as it was identified to be important;
- proactiveness, which is to take initiative actions towards an expected situation, thus sensing a possible problem and dealing with it before, or as soon as, it occurs. Here, trying to learn before the situation occurs is the key point. An example is trying a process and testing it to get suitable back transformation on a kriging, as mentioned by Wright *et al.* (1997); and
- mobility, as there are levels of mobility ranging from weak to strong. It was deemed sufficient to incorporate weak mobility for the requirement of this study, being represented as the communication mechanism in this thesis.

All the above characteristics are implemented for VAC however individual agent in within VAC take advantage of some of all these feature to fulfil their design goal and to achieve the overall goal of fitting a variogram through geostatistical process, table 6.1. shows utilisation of these characteristics by individual agent.

Table 6.1: Agency application on VAC

	Basic Features				Extra Features					
	Autonomy	Reactivity	Social ability	Proactiveness	Intelligence	Learnability	Mobility	Reproductivity	Communication	Data Binding
Data Finder	*	*	*	*		*	*	*	Implemented structure to support Learnability and Mobility feature.	Implemented structure to support Reproductivity feature.
Integrity Checker	*	*	*		*	*	*	*		
Data Analyser	*	*	*	*	*		*	*		
Sampler	*	*	*		*	*	*	*		
Mathematical Modeller	*	*	*			*	*	*		
Strategy Comparer	*	*	*	*	*		*			
Model Fitter	*	*	*	*	*	*				
Model Plotter	*	*	*							
	Function in Jack Engine					Function not in Jack Engine			Program structures	

Mapping these features into GIS and especially Geostatistics shows the potential of seriously moving this technology forward. Once the simulation has been tested with the previously solved data problem and proved consistent, it will be tested on a more complex problem to see its performance.

Software agents provide greater mobility, autonomy, reactivity and flexibility to the service (Bingham *et al.*, 2004). With these functions, they will provide a leverage to GIS by using agent characteristics, including adaptivity, reactivity, autonomicity, networking capability with a sociability and collaborative behaviour, personalisability, pro-activity and cloning ability (Sargent, 1992; Ovum, 1994; Shih, 2001; Callan, 2003; Liao, 2005), and this thesis goes a step further by introducing a reproduction capability. The ultimate ability is to realise the workload and either clone itself (Shih, 2001), breaking down the job into smaller tasks and dividing the tasks among the newly cloned agents, or reproduce to more functional specific agents that will tackle the complexity of data analysis in an unconventional environment.

With the features mentioned here, GIS services, like the agents themselves, will be able to adapt to the user's needs and have the ability to migrate in a self-directed way from one location to another on a specified platform (i.e. a common network system). Also, they can respond to changes that occur in a timely fashion by learning the pattern according to the environment, so that their performance improves over time (Lee *et al.*, 2002). Furthermore, there is a distinct need for improvement in the GIS environment, making the systems more networked and more adaptive to their environment (Longley *et al.*, 2002). The system should have a reactive ability that would allow an interaction of GIS tools to form a social behaviour in a similar environment. Artificial intelligent agents could be used to solve this problem, so the need to look deeper into the collaboration between these subjects (GIS and AI) becomes essential.

The behaviour of the agent, according to its features demonstrated in first in test data and in more complex real life data, the Walker Lake data is presented in Chapter Seven of this thesis.

6.5.1 The Agent Reproduction Capability

Software agents are characterised to be able to achieve communication with both, each other and external users (like humans), harmonising a shared and common understanding of a domain (Lin *et al.*, 2001). Agents have been characterised by many features. However, one significant feature that has not been yet mentioned in agent research but was considered as being important in this research is an agent's ability to reproduce itself:

- **Reproductivity:** this is a feature that is very important, and indeed crucial for an agent which perform functions that have no particular pattern. When an agent realises a change of pattern from the user, it should be able to customise itself to deal with the environment - thus, it should give 'birth' to a new agent having a slightly different capability and goal. The first step is for the agent to be able to

reproduce plans to handle the foreign situation, then if these plans get too complex and numerous the agent should then recreate another agent to specifically deal with them, having goals that will be satisfied with these new plans (i.e. a new agent with slightly different goals but very different plans).

6.5.2 The Agent Reproduction

Agents deal with sampling and modelling of the data to the variogram, such that if a formula to the problem seems to not work appropriately, then an agent could produce new plans, or another agent, and supply it with all the information it has about the situation. Furthermore, the reproduced agent could roam freely to gather extra knowledge to deal with the problem at hand (see Figure 6.20).

Reproduction is 'super-function' of cloning function currently present in agent characteristics. The cloning function is present in most agent software development platform; this is especially true for JACK agent environment which is used for the development of the VAC. Cloning is a mechanism that allows an agent to duplicate itself on temporary basis. However, as described an agent is normally specialized for specific function or task. Thus, in learning new function cloning is not sufficient. Thus it is 'super-function' where a cloning takes place on a permanent basis, see Figure 6.21. This provided 2 main challenges:

1. The agent capabilities need to be easily redefined using external file (xml in this case).
2. Keeping the basic knowledge of an agent distinguished from what it learned through time as offspring need only have this basic knowledge and the knowledge required to handle the new task (function), then it can acquire its own knowledge and own character.

The solution to these 2 problems can be seen in Figure 6.20 which shows the steps (actions) taken when an agent reproduces another agent.

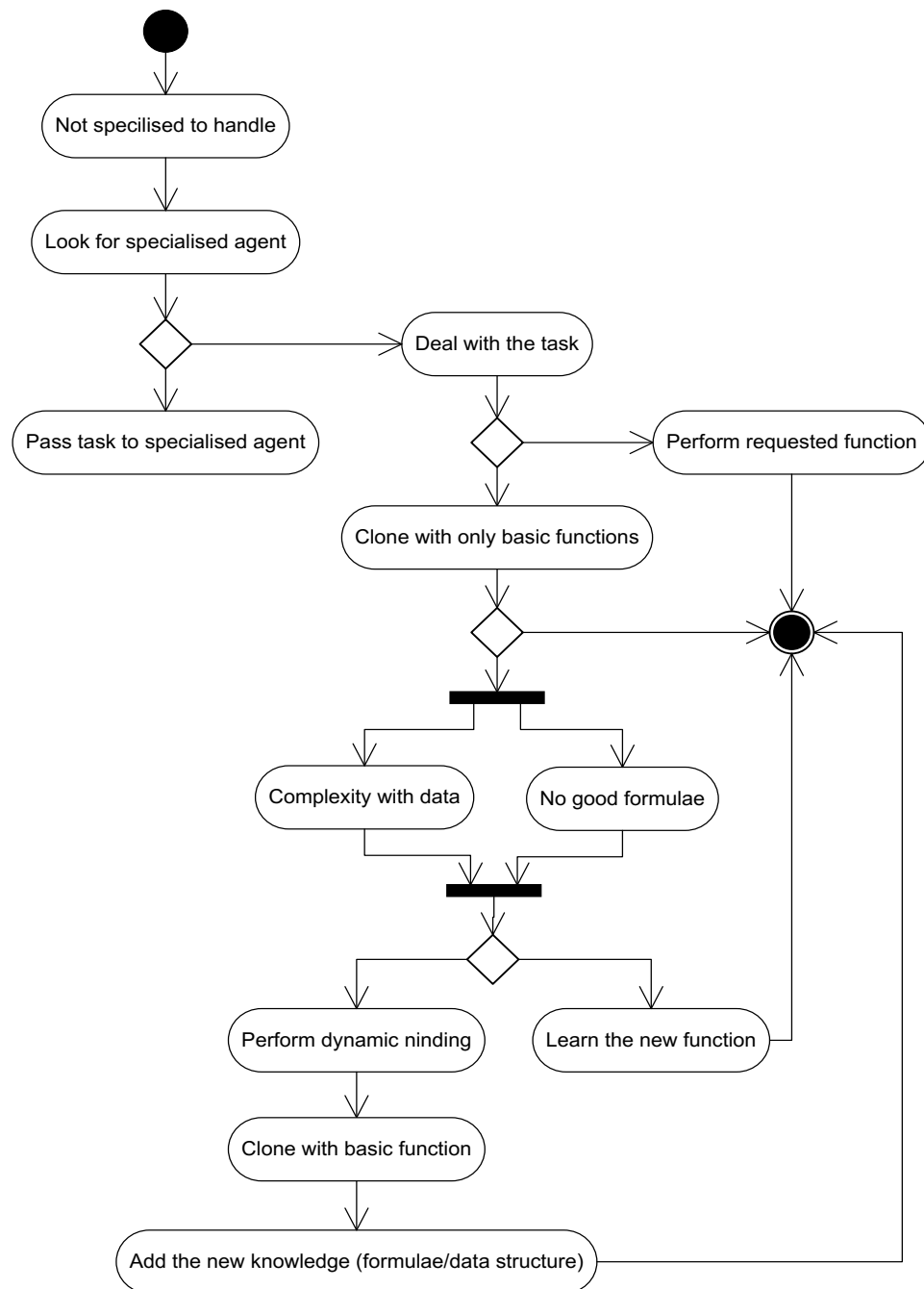


Figure 6.20: Activity Diagram representing algorithm for Reproduction

The algorithm for agent reproduction

```
BEGIN
  New task arrive
  Check if can be done, has capability
  If agent is not capable
    Check its knowledge base if this new capability
    matches the functions available,
    Determine, should it learn
    If yes
      Evoke dynamic binding function
    If no
      First check by evoking communication function if
      there is an agent in the network that can handle
      the task (has the capability)
      If agent Available
        Pass the task to the agent
      If not available
        Using basic clone function to create new
        agent but with only basic capability
        (capability predefined to you before your
        learning start) and a capability with new
        function.
      Instruct the new agent to register to the broker agent and
      its capability registered so next this problem arrive agent
      to reproduce again.
  END
```

To explain reproduction and the need for it, suppose there is an agent 'A1' which already has knowledge about problems 'PA1', 'PB1' and 'PB3'. Agent A1 encounters a new problem which is a manifestation of problem PA1, which we will call 'PA2'. Agent A1 realises that this problem is getting too large and thus could reduce its efficiency. Therefore, it will reproduce a specialised agent for this kind of problem, teaching it the techniques and tricks it had itself previously used (when it encountered the similar problem PA1 previously) and suggest possible solutions to the new problem PA2 (as a learning and teaching process). It will then release the 'baby agent' to mature on its own and be able to later follow this trait of reproduction in its own right. Figure 6.21 demonstrates reproduction function.

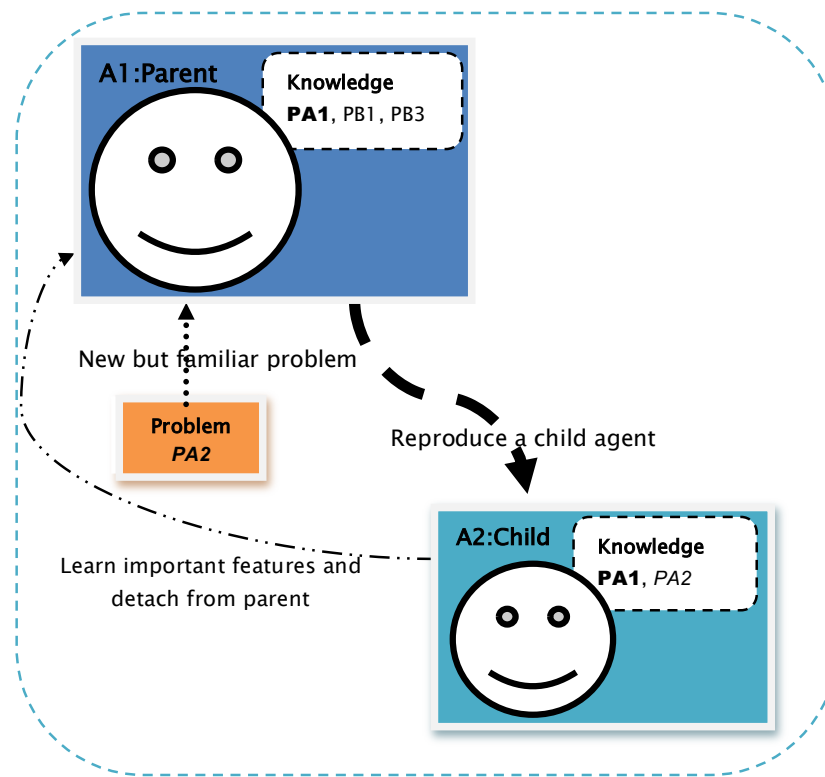


Figure 6.21: Agent Reproduction Mechanism

The need for this feature can be seen, as the two main modeller agents concentrate only on sampling and applying mathematical constructs to the data at hand. They thus will only hold general structures of formulae that would help in fitting the variogram (like the kriging formula). It is explained by Isaaks and Srivastava (1989:7) that “the hallmark of Geostatistics is that each problem has its own customised approach to deal with”.

The general purpose of this feature is to become specialised around data types and applications (thus controlling the size of agents in terms of knowledge carried around for security, agility and functionality). A general example of this is an online market as a specialist agent.

For simplicity to the VAC, the basics of this reproduction feature are embedded as a mechanism whereby an internal or external (expert user) agent could determine a new

functional technique to solve a problem that was not able to be correctly resolved previously. This structure allows a new mathematical formula to be passed as an argument, if the final results were not satisfactory, to either the 'dependee' agent or to the human expert - this feature is termed 'the dynamic binding mechanism'. In this thesis, only the human expert input is regarded, while the function to incorporate agents that will handle this function is kept as an improvement to be made in the future, as it would take a considerable amount of time for agents to be able to learn and reach the required level of intelligence.

The agent will be 'born' with minimal knowledge so it can do simple functions and, like a child, become inquisitive and compare its own answers to that of the expert. Here, the term 'born' is used to differentiate the intelligence being introduced into the system from other pieces of software. Due to the expert's input and agent's own observations, in time it should be able to become more intelligent, such that the process could work the other way round and the expert is able to raise their query and the agent can suggest possible models. Of course, this is not the desired effect of the Variogram agent, but it will facilitate the learn ability of the agent.

The dynamic binding mechanism is explained in the next section and its experimental results will be presented in Chapter Seven.

6.5.3 The Dynamic Binding Mechanism

In GIS, every problem (associated to the data) has its own way of being solved. An example of this concept is having data to be studied which has spatial trends. These trends can either be removed, or left unconsidered when undertaking an analysis, depending upon the nature of the data. If the data is about the growth of fish in a lake, an important factor is simply the number of fish, while if this same data was dealing with iron ore, the clusters would be important. Thus, the data also needs to be determined according to its qualitative nature and not just its quantitative nature.

This phenomena needs to be determined during the run time of an agent. To achieve this, the reasoning process will be imported into the agent's plan during the run time. This concept is shown in Figure 6.22.

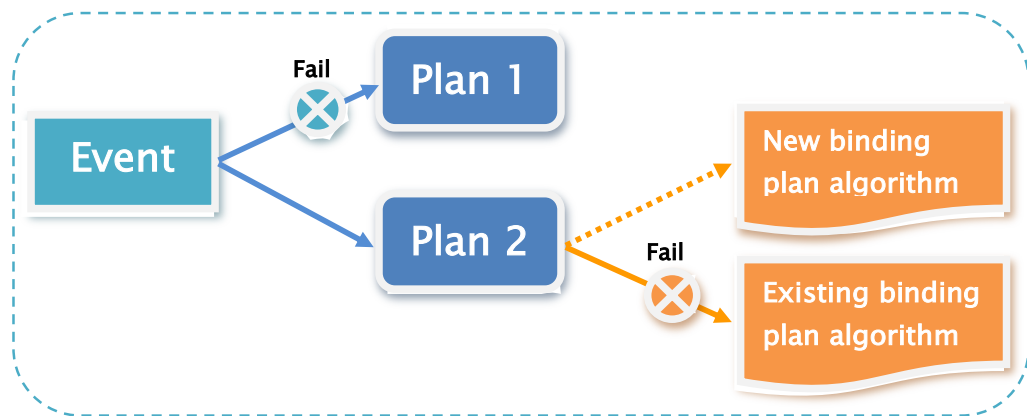


Figure 6.22: Structure of Dynamic Binding of Plans

At this stage, the human expert too could communicate with the agent, for example to tell the agent that it has categorised the data wrongly. This is achieved by an agent being able to provide a pop-up which would request a new formula to be used for the analysis. Consequently, the agent would re-examine the original data and re-process it. However, at this stage, the other agent that had the previous run data sent to it would not stop executing and so the process would be as if there are two separate data types being processed. This would not affect the system processors, as the agents work autonomously to each other, and mostly on different machines.

A specialist console is designed to for the VAC to help binding formula, Figure 6.23

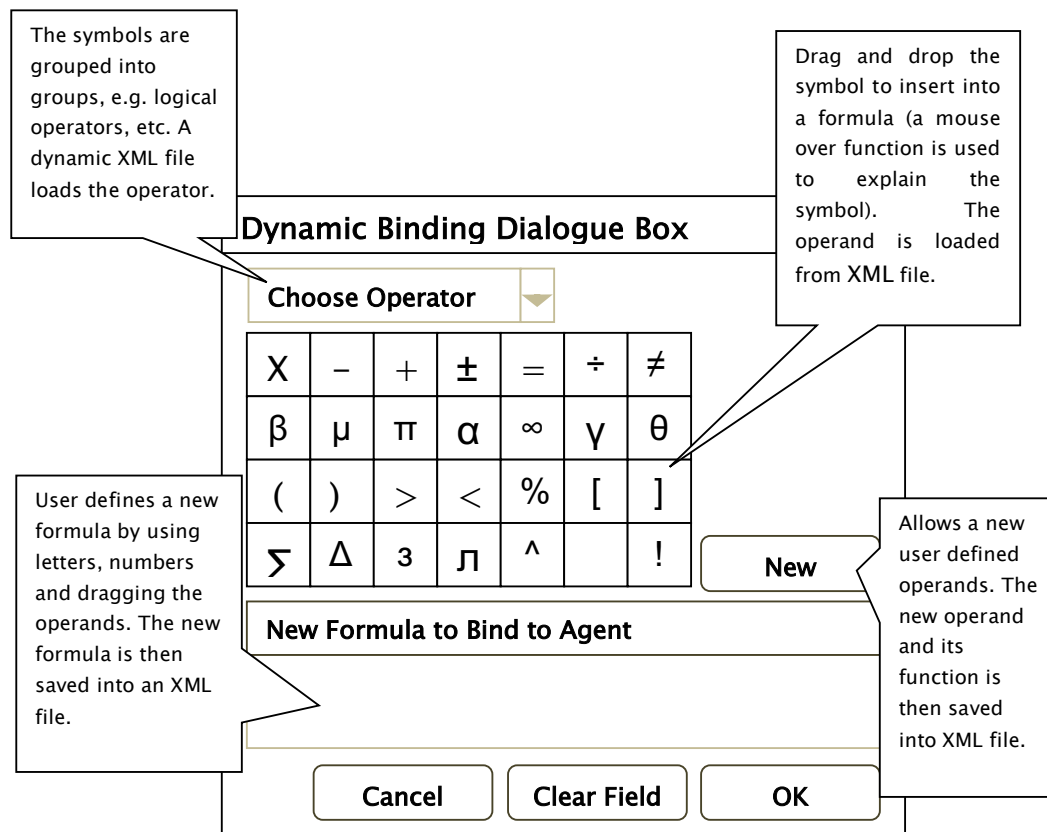


Figure 6.23: Dynamic Binding Console Design

This function introduced a challenges of which were partly handled (the main requirements) and partly left as future work.

1. The ability to insert new mathematical symbols; as some formula might require symbol. An experiment of inserting a square is given in chapter 7 by inserting a symbol and define is being a float number x, by expressing it as xXx , where X is multiplication sign. This was simple enough and easy to enter but when dealing with complex symbols like Σ which means the sum of all and in turn requires a *programmed "for loop"* the implementation becomes complex. And thus the system will accept and symbol and its definition and it reject what it cannot understand while giving a option to remove symbol to other agent if the symbol constantly yields wrong results.

Because of the above given problem they mathematical symbols are exhaustively inserted during the development of an agent to support their function. Furthermore, the symbols can be updated directly in the xml file since it is where all the symbols and their functions are store and always compiled as in class function during runtime of an agent.

The idea is to progress this console to eventually be able to handle natural language (semantic input). Currently VAC can only bind formula using action flow (activity) given on Figure 6.24 and the algorithm that follows.

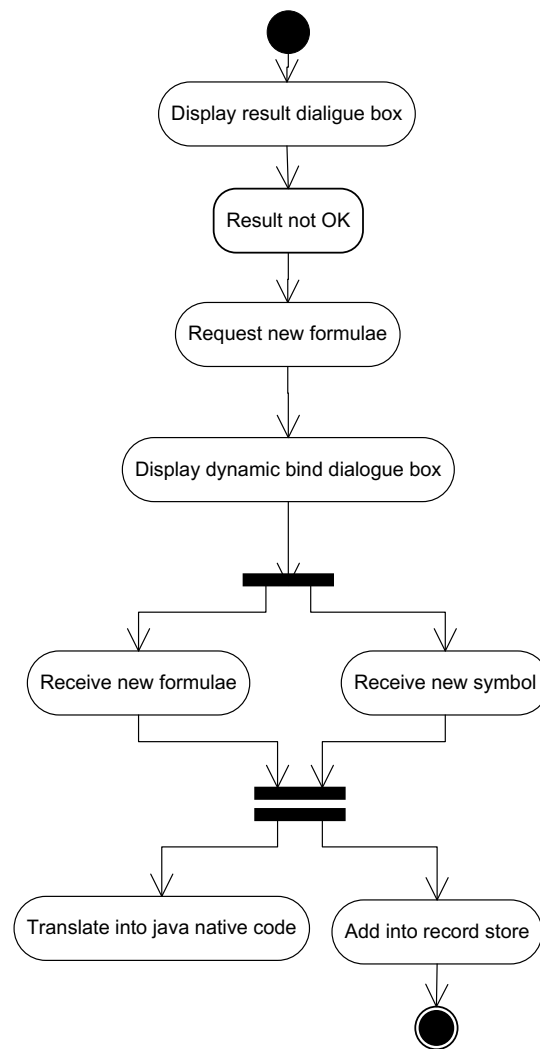


Figure 6.24: Activity Diagram representing algorithm for Dynamic Binding

The algorithm for dynamic binding

```
BEGIN
  A new plan is required and the console to bind is called
  Get new formula
    Add new formula and its explanation to xml file
    according to circumstances it was developed //e.g.
    //during trend analysis of data type 'sample1' the
    //normal linear regression was rejected and new
    //formula: t-test was created the formula is ""
  Use the formula on the current data
  Confirm user acceptance to the results
    If results are good
      Tag the formula good for circumstance
    If results not good
      Retain formula and provide it as a plan
      But display for scrutiny to other users
END
```

6.6 The Agent Communication

A communication mechanism is required to handle large datasets and/or when data arrives which is unfamiliar to the given agent. An agent has the knowledge that there is another agent within the network that has similar capabilities to itself, but which might have different knowledge and be able to handle the different dataset. The stress here is on the word 'might', as an agent does not have a definite idea as to whether another agent that has the same function as itself will also have different knowledge. The functional mechanism of the architecture is only hinged, and so relies upon a look-up table in the Broker agent.

To ensure accuracy of functionality on ad-hoc systems communication is facilitated by a broker agent. A broker agent is none integral part of VAC, as an agent specifically design to assist mobility and communication for the in within VAC environment. The inspiration to this function is due to the lack of sufficiency of mobility feature in JACK Agents platform. The communication function is given in figure 6.25.

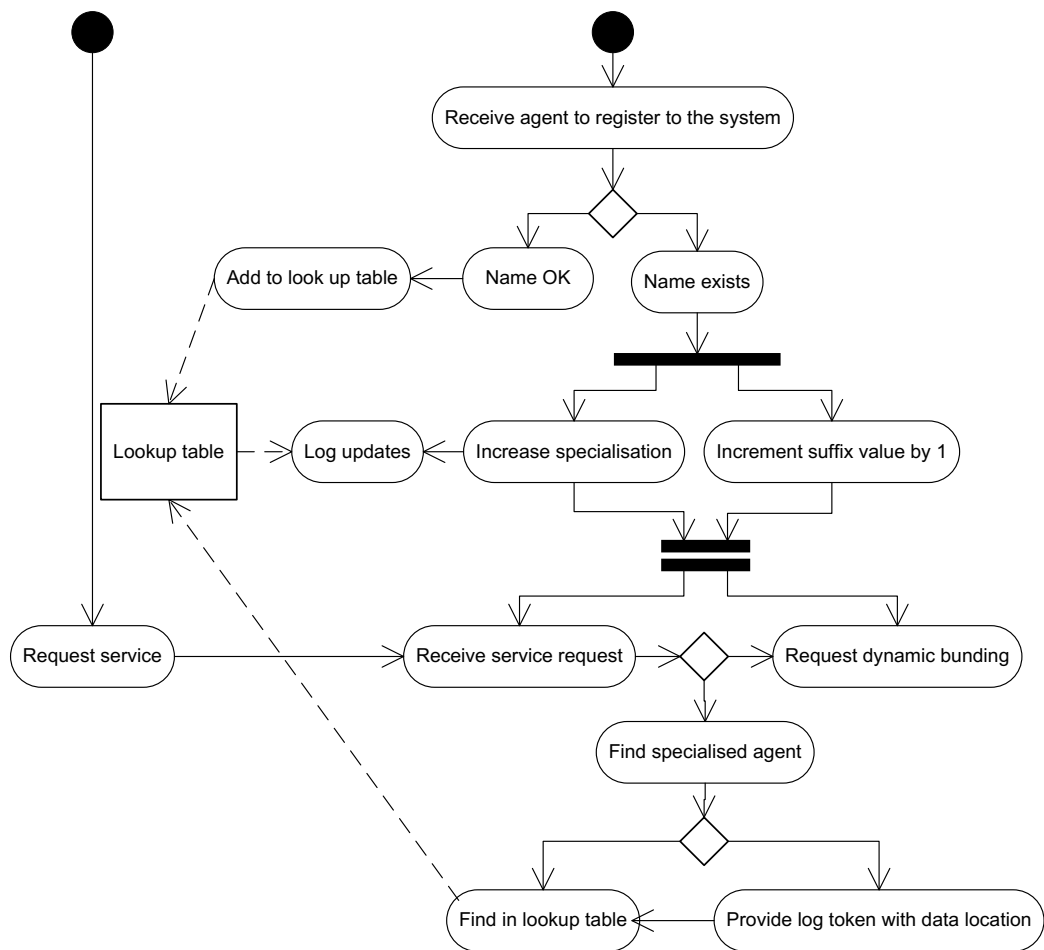


Figure 6.25: Activity Diagram representing algorithm for Communication (Broker)

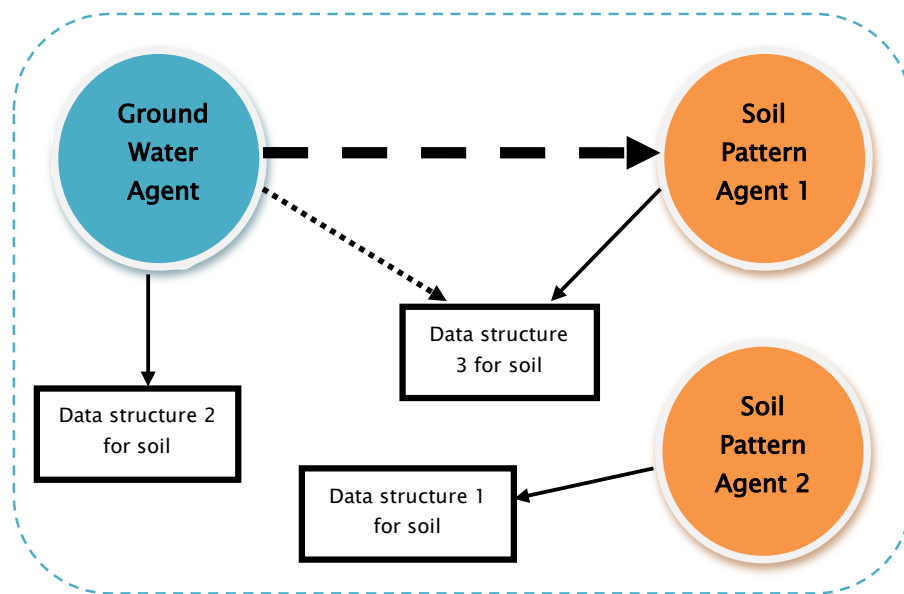
The algorithm for communication

```

BEGIN
  Agent first run in a machine or network
  Register to the broker agent of the network - your name -
  main functionality - your machine address (it is your
  domain and network address mainly IP, agent-name@address)
  Stay on waiting state for request
  CASE 1: WHEN Agent is invoked by a requesting agent
    Provide your functionality according to your
    capability
  CASE 2: WHEN providing function and realize you are
    not specialised visit the look up table and see an
    agent with similar function as yourself
    Send a request for help to broker agent
    When receiving an acknowledgement- send a token
    with the data location the agent should act
    upon and what task should it be.
  Send a communication termination from the requesting
  agent and point the request agent to the contacted
  agent
  Terminate and stay idle-waiting
END

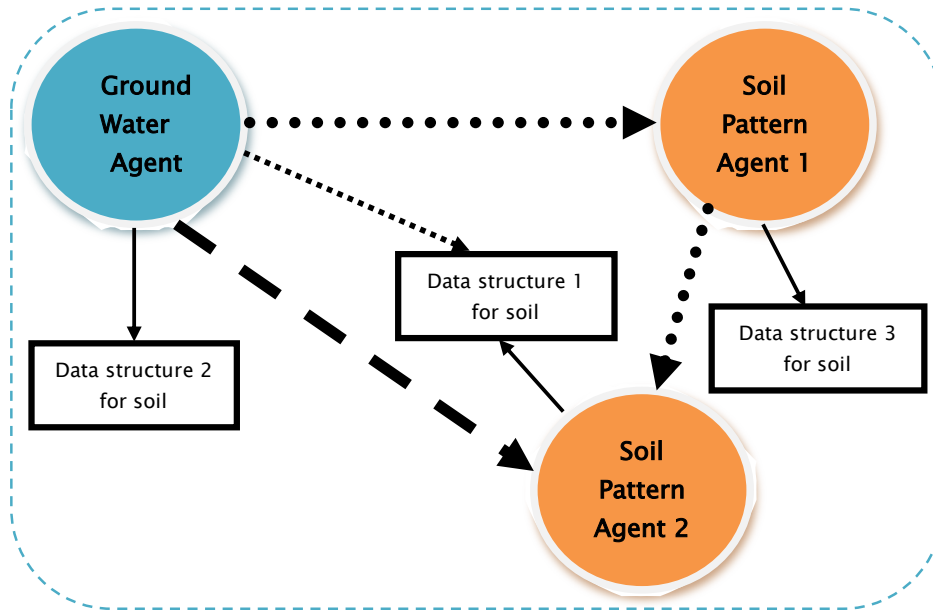
```

To explain the mechanism the following example is devised. The structure is such that an agent specialising in ground water levels (termed 'Ground Water Agent'), which hypothetically let's say specialises in dealing with smooth, gradually varying trends. This agent needs to access information from an agent specialised in dealing with soil pollutants (termed 'Soil Pattern Agent 1'). Hypothetically this agent is specialised in dealing with pits and peaks that could be quite sudden in variability (see Figure 6.26a). However, it happen that this agent being contacted is not experienced enough to solve the problem and needs to contact another soil pollutant agent (termed 'Soil Pattern Agent 2') to solve the problem (see Figure 6.26b). A test of the main requirements of this feature is provided on chapter 7.



6.26a: Communication Flow Without Utilising VAC Functionality

So, the hydrology agent 'Ground Water Agent' can access data from Soil Pattern Agent 2 through Soil Pattern Agent 1, as shown in Figure 6.22b below:



6.26b: Communication Flow Utilising VAC Functionality

As the intelligent agent provides the ontology (explained in chapter 3), the described structure works by giving an agent the semantic of the other agent. If an agent encounters another agent with a semantic that it does not understand, it should then request help from a mutually understanding agent to acquire the capability to handle those semantics, or even to get the task done.

Ontology already contributes immensely to the current trends of geosimulation. However, the full power of ontology has yet to be fully realised and it could indeed be made to be much more effective. Lin *et al.* (2001) suggest that ontology should be domain specific, but what they don't show is how the specific domains can interact with different domains when needed. Through the given communication mechanism (Figure 6.22), this can be achieved through VAC, which aims to build a domain-specific ontology

that can form an interface when needed to communicate with external agents. The actual experimental runs can be found in section 7.5.

6.7 The Agent Learnability

An agent needs to be able to log the geostatistical tools (formulae used within a plan) used by it when formulating the decisions it has made. These logs can then be reported to a human expert, so that the expert can point out any mistakes and critically assess if the decision is good enough or not. If it is not good enough, then the expert needs to suggest a better decision, so that the agent can work the decision in reverse in order to try to ascertain at what stage it made the mistake and rectify the log at that point. This mechanism will rapidly expand the knowledge of the agent and allow a much better relationship between the human expert's thinking mechanism and the agent's thinking mechanism.

There are also other methods that are used to rectify a decision, achieved by the VAC checking on itself. This is done by using its previous logs. If the results are expected to trigger certain actions (for example, producing a certain graph, table or decision) but this action does not take place, then the agent can reverse work from the time of the expected action, so that the point which caused the problem can be noted into the log. It is important that, at this point, the agent can receive feedback regarding the action, so that it can decide what was wrong with the decision (i.e. whether the decision was wrong, or was it just a different phenomenon with similar data characteristics to other previous processes). This helps to fine tune the agent. Figure 6.27 shows the learnability of VAC.

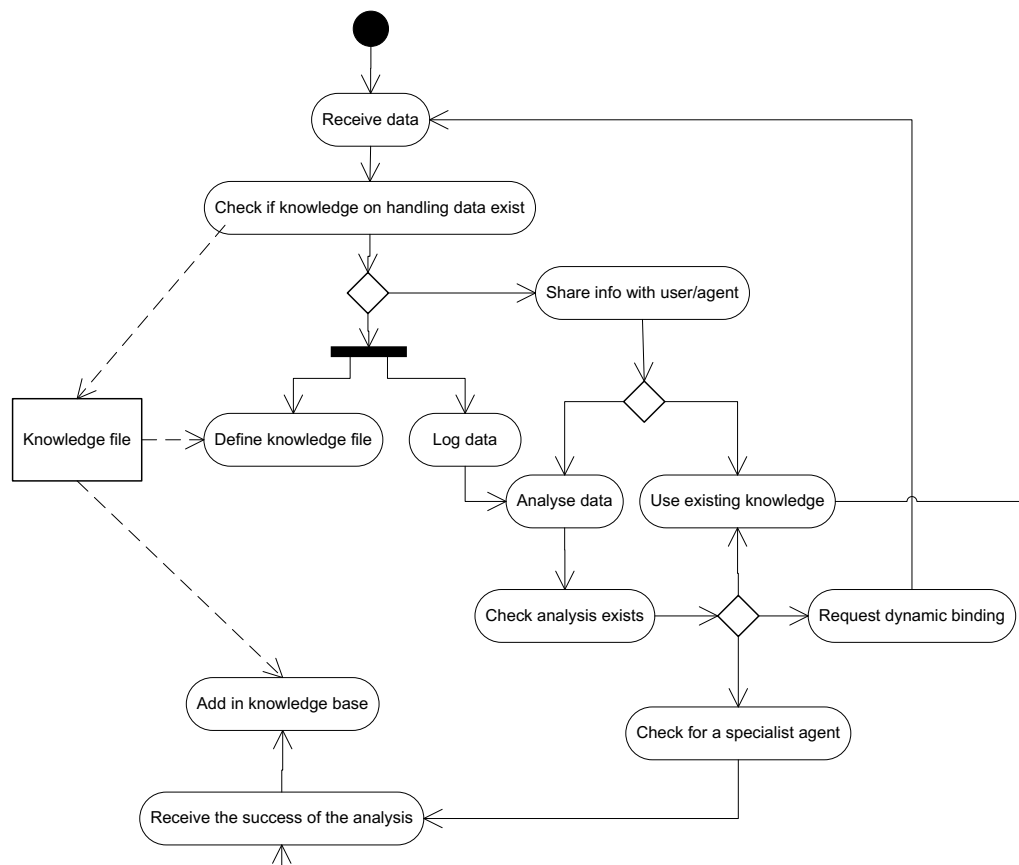


Figure 6.27: Activity Diagram representing algorithm for Learnability

The algorithm for agent learnability

```

BEGIN
  Receive data
  Check if data type has been handled before
  If yes
    Use the agent experience for the algorithms, formulas
    and process
  If not
    Check if expert agent on this problem exist
    If yes
      pass the problem to expert
    If not
      Request problem definition
      Describe the data (file format,
      representation format and interact with
      user (human or agent that requires the
      results) while logging every step of the
      acceptance results through analysis
      process)
END
  
```

To point out the direct implications of the expression worked above, the relationship between the support and distribution of the data for a spatial process will be described as explained by Atkinson and Tate (2000) will be used as an example. Atkinson and Tate (2000) express that it is important to evaluate the support of the sample data and the support of the intended final estimate. If these two are different, then we will need to deploy the dynamic binding mechanism so that the support for the process at large can be derived. This structure is required for the many problems that exist with estimation tools (e.g. if the decided tool for the process is such that the mean is kept constant and only the variance is changed) (Isaaks and Srivastava, 1989). This is due to the fact that, in this situation, it is arguably better to do some form of correction than not do anything at all.

In terms of supporting the intelligence theories chosen on this algorithm in content of the machine, if to think like a human (i.e. have intelligence), it is important that it also behaves like a human. Thus the human behaviour is needed to observe a decision and, if the result is not satisfactory, rethink the decision and add the current decision as a (negative) weight to what it already knows, so as to get a better shot at making the second decision. At this moment, one might ask “so how can one trust a machine to make decisions on a critical problem if it has the probability of making wrong decisions?”. Looking at the work of Alan Turing, the father of artificial intelligence (quoted in Hodges, 1983), he states that “if a machine is considered infallible than it can never be intelligent (*sic*)”. To explain this point, first let us look at the real meaning of intelligence (or rather, most perceived meaning). Gottfredson defines intelligence as:

“...a very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience” (Gottfredson, 1997: p13).

This explanation refers to the ability of general comprehension when making decisions which, of course, can go wrong.

“...it reflects a broader and deeper capability for comprehending our surroundings - ‘catching on’, ‘making sense’ of things, or ‘figuring out’ what to do” (Gottfredson, 1997: p13).

Furthermore, Piaget's and Vygotsky's theories express that intelligence does not mean getting it right all the time, but simply the ability to learn and make decisions in a given situation (APA, 1996). The decisions could be right or wrong, according to the perception of the situation. Furthermore, Bratman (1999) argues that if an agent satisfies the BDI mechanism, then it can satisfy the intelligence notion. As the JACK programming platform is being used to develop the agents used in the experiments of this research, and JACK is identified as an agent platform supporting pure BDI (Wooldridge, 2000), then the argument is held that the agents here are intelligent agents.

6.8 Conclusion

Here we have defined the structure of the agent system using the Tropos model. There were some limitations identified in the implementation stage, along with the need for a non-software development expert to understand and contribute to the final system. To deal with this issue, a new modelling mechanism for presenting the implementation stage of the system was identified and used as the overall system architecture. The results of this architecture are presented in section 6.3 of this chapter. Furthermore, a new agent feature was identified through the development of this thesis, that of agent reproduction. The complexity of this feature is explained, then referred to against the existing literature review for comparative analysis. However, the reproduction mechanism, as suggested, is identified as being too complex for the level required here, and thus is explained in three phases:

1. The cloning mechanism as reproduction; this is an existing feature in a software agent which creates a copy of the original agent.

2. The dynamic binding of plans as reproduction (as these new plans subsequently change the agent's functionality) - the agent with new functionality then acts independently.
3. The actual reproduction mechanism; as having one agent 'reprogram' another smaller, more compact and agile agent.

The dynamic binding mechanism, as novel agent architecture, is used throughout the experiments in this research and the results are presented in section 7.4.

These two contributions to software agent design and functionality were deemed to be necessary for software agents to be utilised as service providers in GIS.

CHAPTER SEVEN: RESULTS AND ANALYSIS

7.1 Introduction

This chapter looks into the functionality of the variogram agent system by validating the data it analysed. This validation will be conducted by examining the internal mechanisms of the system and the final results it provided, which will be defined as external validation. This will be explained with respect to Isaaks and Srivastava (1989) who have worked with this data, showing conclusive results. Furthermore, the functionality of the VAC will be compared to other GIS tools that offer agent-based services and are available to geostatisticians. Also in this chapter, the new theories and features that were introduced in section 6.4 for agent-based GIS will be analysed and examined.

The starting point for internal and external validation will use geostatistical analysis for the variogram construction process (detailed in section 7.1.1 and shown in Figure 7.1). The problems encountered during validation will be outlined and their solutions discussed.

7.2. The Agent Functionality Test

By 'internal', it is meant that the system will be validated in terms of its process flow, the code functions and programmatic (semantic and syntax) errors. The important point to note here is the ability for each individual (internal) agent to reach its desired results and final goals. A cluster analysis agent should be able to analyse the clusters and provide results, so they can be passed on to the next agent in line. Thus, the experiments will be undertaken without considering the data or its analytical results, only focusing on the fact that the agent is handling the data appropriately, as designed. If the agent fails to do so, it will be determined as an error (programmatic), which will be investigated and explained either with the appropriate solution or deemed that it cannot be solved. The first step will be to look into the Figure 6.11 to explain the process that takes place during agent analysis, shown in Figure 7.1.

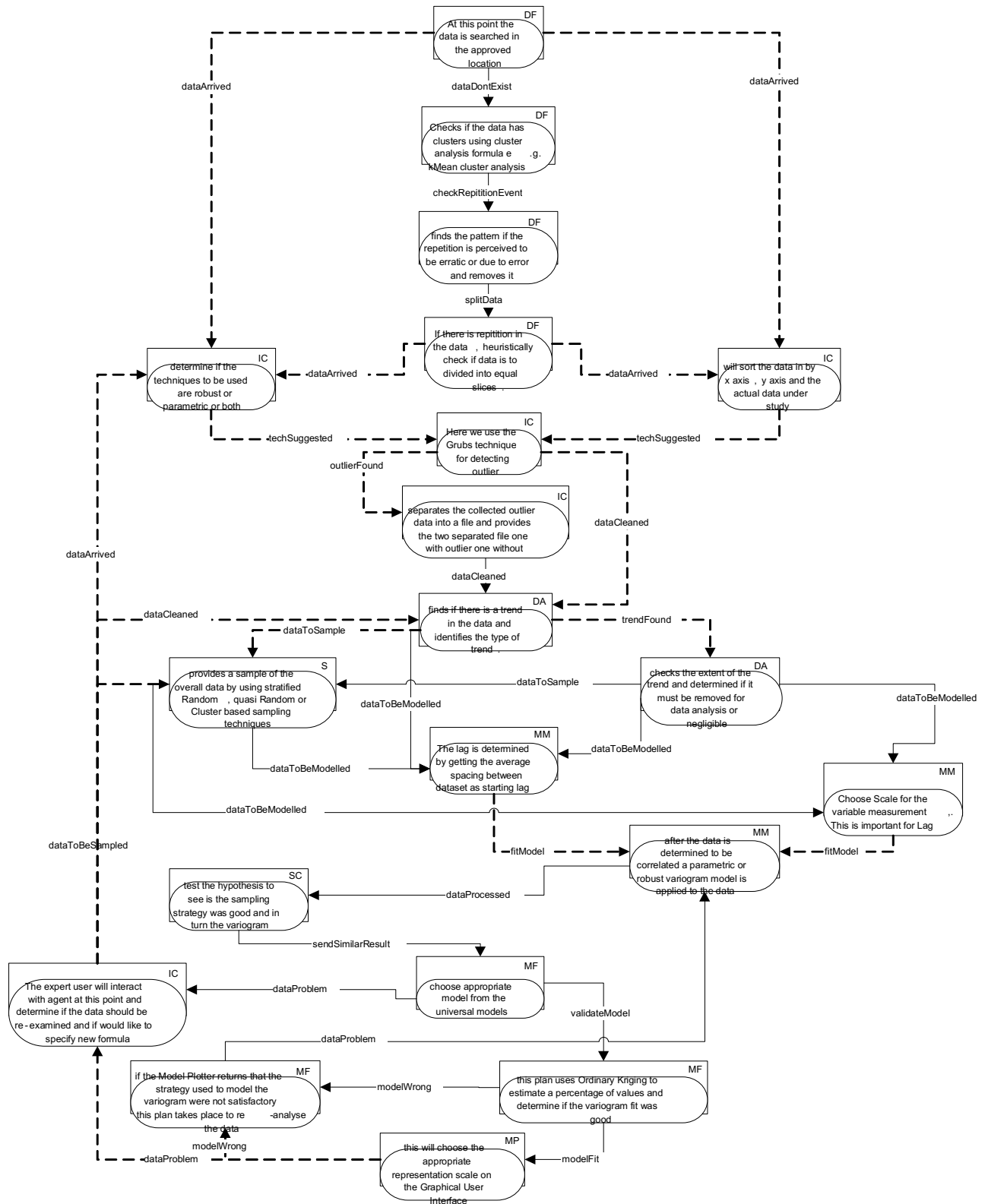


Figure 7.1: Process that Takes Place During Agent Analysis

The process is started by a human or software agent placing the data file in a designated location (computer folder). The agent responsible for dealing with data at this stage is the Data Finder Agent (DFA), using its capability searches for a specific named data file in an approved location. After this step, the DFA will check if this data (as exact data or a similar structure) has been dealt with before. If 'yes', it will use one of its plans to alert the Integrity Checker Agent (ICA), otherwise it will determine the structure of this data, which it will add to its knowledge. The determination of the data type is based upon the agent checking whether the data has clusters, using cluster analysis formulae (e.g. quadrat cluster analysis), examining repetition patterns on the data if available and perceiving whether it is erratic or due to errors in the data file, then dealing with the error (by removing the perceived problematical data).

If there is repetition in the data, the DFA will heuristically check whether the data should be divided into equal slices or use cluster division for sampling. After this step, the DFA will reach the dataArrived goal and the process will be taken over by the ICA, which will have to determine if the techniques to be used are robust, parametric or both, and sort the data by x axis, y axis and the actual variables under study. When the data has been sorted and the technology suggested, the ICA will trigger the techSuggested event, which will determine if there are any outliers (using the Grubbs test technique). If it finds any outliers, it will call up a plan to clean the outliers. However, determining outliers is never a precise exercise (Isaaks and Srivastava, 1989). So, even though the outlier data is removed, the plan will separate the collected outlier data into a new file and provide two separate files - one with the outlier and one without. Once the data is found to not have outliers, or to have had outliers but these are now removed, then dataCleaned will be initiated and the Data Analyser Agent (DAA) is instructed to start its function.

The DAA attempts to find out whether there is a trend in the data and, if so, identify what type (linear, quadratic, etc). If not, it will pass the activity to the Sampler Agent (SA). However, if a trend is found, the DAA will check the extent of the trend and determine if it must be removed for data analysis or whether the trend is negligible, whereby it will

pass the activity to either the SA or Mathematical Modeller Agent (MMA). The MMA will handle this data by either determining the lag or by setting the average spacing within the dataset as the starting lag and choosing a scale for the variable measurement, which in itself is an important step in choosing lag space. The results of both plans will be used to help determine the technique for modelling the experimental variogram (determined as the data is correlated, so a parametric or robust variogram model is applied to the data). In some circumstances, depending on the agent's decision from the data analysis, the SA will be called to provide an interim step before the process is taken over by the MMA, which provides a sample of the overall data by using stratified random, quasi random or cluster-based sampling techniques.

After this step, the process will be passed to the MMA, using similar plans to those it could have passed when coming directly from the DAA. The MMA will, in both cases, pass the process to the Strategy Comparer Agent (SCA), which tests the hypothesis to see if the sampling strategy was good and, in turn, the variogram. Then, the Model Fitter Agent (MFA) will find the appropriate model from the universal variogram models. At this stage, the chosen model will be validated by the ICA and the MFA's own internal plan, which will check the goodness of fit using techniques (e.g. ordinary kriging) to estimate a percentage of values and determine if the variogram fit is good. This process will end with one of two outcomes: (1) if the strategies used to model the variogram were not satisfactory, then a plan provided by the MPA is initiated to re-analyse the data; or (2) if the fit is good, it will then choose the appropriate representation scale on the Graphical User Interface, then send the process to the Model Plotter Agent (MPA). The MPA itself might decide that the plan to re-analyse the data must be taken. The ICA will allow, using dynamic binding, the expert user to interact with the agent and determine if the data should be re-examined, and whether the expert user would like to specify new formulae which would determine where the process needs to be restarted from. The choices are the DAA, ICA and SA. The process showing agent interaction and plan functionality was provided earlier (see Figure 6.17).

The following experiments are done on PC running window 7 with 3.2GHz processor, 8 GB RAM with a dedicated storage space of 10 GB specifically for the VAC. Basic tests (memory over flow) have been done to determine that the agent is able to realise its capacity and able to distribute overflow process (the agent divide the task before hand and send request for help) to other computers. This effect will be shown during communication experiment on this chapter. Since the agents are able to determine their capacity there will be no issue on validating the data volume but to determine the available capacity and be able to efficiently divide and distribute the sub-files.

7.3 Internal Validation

The internal validation process identified a number of programming errors. We will now look at the programming problems and how they were corrected. In this low level analysis, there were two major issues in the functioning of the agents that needed to be dealt with separately:

1. the non-response problem, and
2. the large dataset problem.

7.3.1 Non-response Problem

The main issue here was the agent becoming non-responsive. This was determined to be caused by one of these two factors:

1. data not being located and causing the system not to execute at all; or
2. agent pausing and not passing the data to be dealt with to other agents.

These issues can be seen in Figures 7.2a and 7.2b respectively. The major problem in this case was that the agent could stop working and not do anything at all. This was found to happen if the data was not available. By not available, this could mean that:

- the data is missing and not available, which could also happen if the data was misplaced (as shown in Figure 7.2a);
- the data name was wrongly spelt and not able to be found; or

- the data being called was in the wrong format.

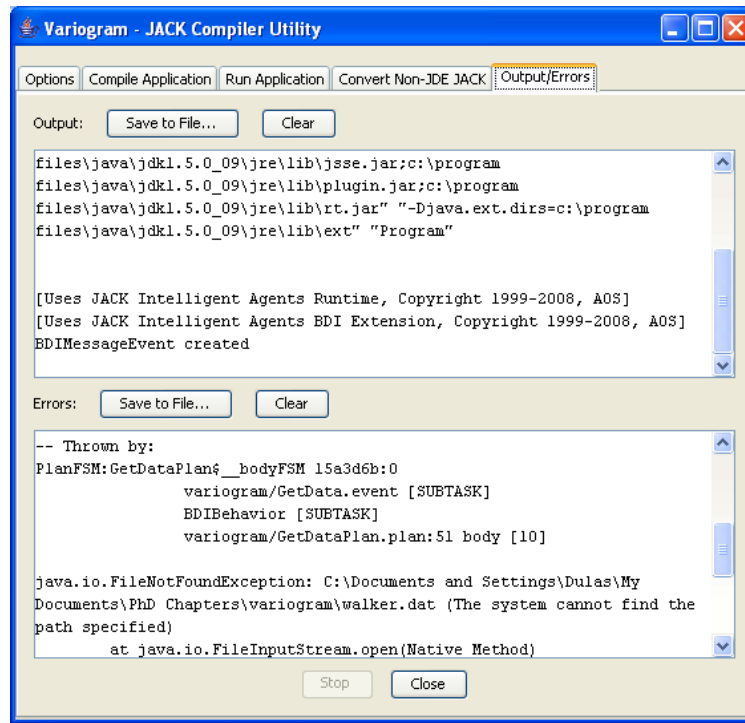


Figure 7.2a: Log Showing Message for Misplaced File

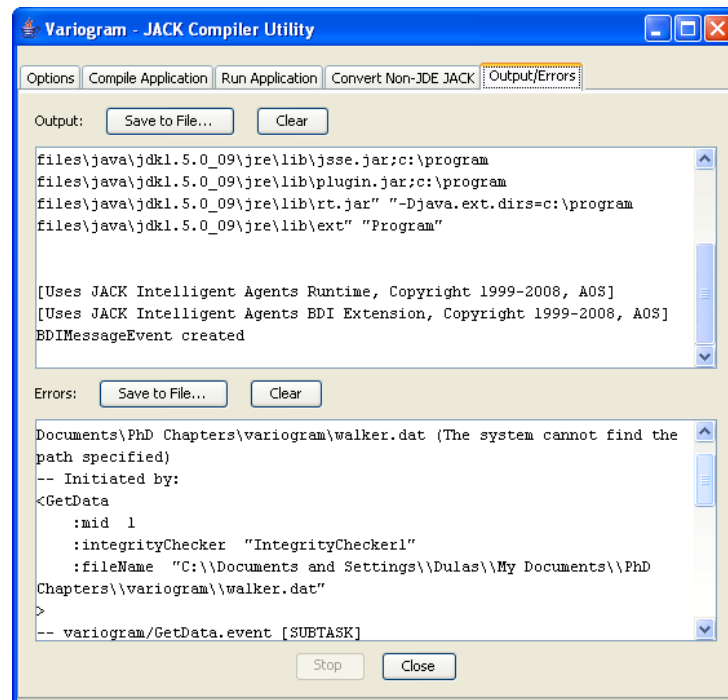


Figure 7.2b: Log Showing File That is Non-responsive

Figure 7.2: Logs Showing Internal Errors of the Agent

In Figure 7.2a, the bottom window of the JACK compiler utility shows that the Integrity Checker could not find the Walker Lake data (Walker.dat), even though it was notified that there was data in the directory. To find whether the problem was in fact that the data was not available, data was physically placed in the directory, but the error still persisted. After doing 'white box' testing, it was found that the address of the directory had changed for the current agent. This was fixed by recoding the data address.

This problem arose in three runs during the experiments. This showed the agent to be unreliable and more research (as shown in the paragraph above) was conducted to tackle this issue. Currently, the agent programming language cannot track the data, so in this case a technique was devised and the data location was specified in three different locations to aid the user should it have forgotten the correct location.

The agent also became stuck at times. On a number of occasions the agent became stuck when checking for clusters or determining trends, when the system seemed to get into an infinite loop. Figure 7.2b shows the non-responsive problem of data not being available from the start. Four runs were executed, with all returning the same problem (same log). These runs were to test the agent's behaviour when:

1. the data was not present at all on the called domain;
2. the data was available, but the name of the data was intentionally misspelt;
3. the data was available, but with an unknown file extension name; or
4. the data was available, but in the wrong format.

Problems 1, 2 and 3 were not solved, because there were no implementable solutions. The only solution was to make sure that the data sending agent (human or software) put the data in the right domain and in the right name. Problem 4 was identified as being important to be solved to facilitate the agent intelligence feature. This was done so that an agent, when it realised that the data was in a format that it could not understand, would find another agent in the network that could understand it and ask for the data.

Thus, this function was solved using the intelligence and communication features of the agent. The intelligence feature determined that the data was useful but in a format it could not handle, then it utilised the communication feature to ask for help from a nearby agent that could handle the data (this is discussed further in section 7.5). It could also utilise the agent reproduction feature, through dynamic binding (shown in section 7.4), if it did not find any agent in the network that could deal with this data.

7.3.2 Large Dataset Problems

The exhaustive Walker Lake dataset caused the agent to make a bad data segmentation decision which caused the VAC to produce wrong results, which consequently made it hard to choose the final best fit model. This occurred due to the memory requirement for reading over 1000 records, then analysing them. At the time of the study, the VAC was used on a peer-to-peer network of two computers and perhaps if the Broker agent had had a larger peer-to-peer system to work over this problem might not have occurred. Furthermore, a process like ordinary kriging involves regenerating the 1000 record dataset, multiplying this to over 10000 values. In the case of the 447 Walker Lake data, 330 were sampled, which generated 8772 kriged data. This caused an operating system memory overflow which subsequently crashed the whole program. Figure 7.3 shows this problem being flagged by the ESRI ArcGIS tool.

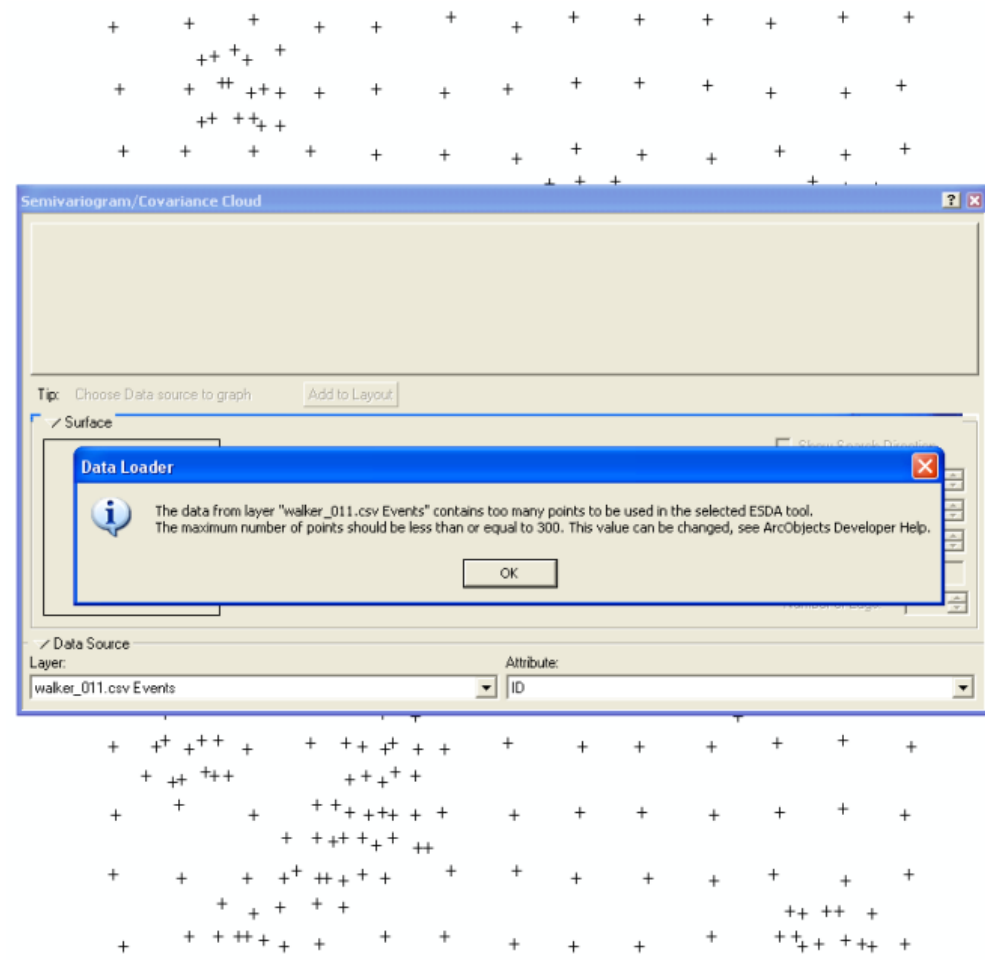


Figure 7.3: Analysis Being Rejected by ArcGIS due to the Dataset Size

This issue is tackled and helped by examining the distributed function of the agents. The original design was that an agent would provide a number of possible variogram models and suggest the most appropriate one. Then, once affirmed by the expert, it would save this knowledge for the next time. This is facilitated at the start of the agent analysis, where the agent will check whether this kind of data has been dealt with previously. This will also check for data similarity as well. If the data is almost the same, the agent will use the exact same model agreed and suggest any very close ones that should also be looked into (for better analysis or if the first suggested model is not workable).

However, a problem arose here as each of these models had to be tested with more than one lag, which created the exponential problem (the more models possible, the

more lags possible and so forth). Thus, the agent should either find another agent on its range that can do the analysis or reproduce a more specialised version of itself and divide the task. The agent can do either, but as passing the data to another agent is easier, it will try this first and, if not possible, it will reproduce itself and instruct its newly reproduced agent to define four possible lags within a given range (which the original agent will avoid). The communication mechanisms of the Broker agent are explained in section 6.5 and the experimental results are presented in section 7.5.

7.4 External Validation

After determining the internal validation issues the external validation test were conducted. External validation will be ascertained through the process of examining the outcome of the agent results. Here, the final results will be assessed and compared to conventional geostatistical tools. The accuracy and usefulness of these results will be determined and problematic areas of the agent system, if any, will be identified. After all the required data handling mechanisms for geostatistics have been completed, a variogram will be constructed and compared to that published by Isaaks and Srivastava (1989). Finally, the performance of the agent system as a service provider (the VAC) will be compared with other agents in GIS that are determined as service providers by their respective authors.

The first concept for experimentation was the data handling mechanism, being that access to data is an essential function for the agent to run at all. Thus, the experiment was undertaken on data placed by human, other proprietary software (other established software used by human experts to help dealing with data analysis) and software agents as users. The concept of using other software systems or an external expert user is what prompted the requirement of the dynamic binding feature for the agent. This feature was explained in more detail in section 6.2.1. This problem escalates to ultimately crash the agent, which brings us to the next important feature which is that the agent needs to be able to save processes through the data. This is achieved through

the data integrity function, which is an internal rule of any good software development mechanism, including both object-oriented and agent-oriented. However, in this case, it is further enhanced by the agent defining an internal structure which not only keeps the data in a consistent state (hence saving the data after each transaction), but it also provides a log that tells the agent where the last process was and what path it took to get there.

The first line shows the last process, the other lines show the process in their order of appearance as to what took place to get there. Because of the heterogeneity of problems that GIS is employed to solve, it is impossible to programme agents for every eventuality. Therefore a dynamic binding mechanism is required.

7.4.1 Scenarios: Testing the new agent features on VAC system

The dynamic Binding and Communication functions are considered to be two of major novelty of this thesis, thus will be used as the central point of the experiment setup for the system validation.

Two scenarios are designed to specifically test these new functions. However, the scenarios will also serve for the observation for the functions described on chapter 6. For these scenarios four data files with good structure (for the test) were created with intentional inclusion of clusters, trends and are known to fit a standard variogram model. Two more files were created one with no data and the other with corrupted data.

7.4.2. Scenario 1: Testing Dynamic Binding and Reproduction functions

Using the dynamic binding function, the agent system will provide a unique mechanism into GIS. This will create a knowledge sharing mechanism between agent and users which will help to deal with data. It will enable data to be analysed by interacting with the user and providing possible results, from which the user can determine the best possible choice. This is unlike the currently available systems, which simply provide calculation algorithms that the user can choose from and, if the result is not satisfactory, the user

will have to repeat the process again until the result is satisfactory. This scenario follows the following reasoning algorithm (see Figure 7.4).

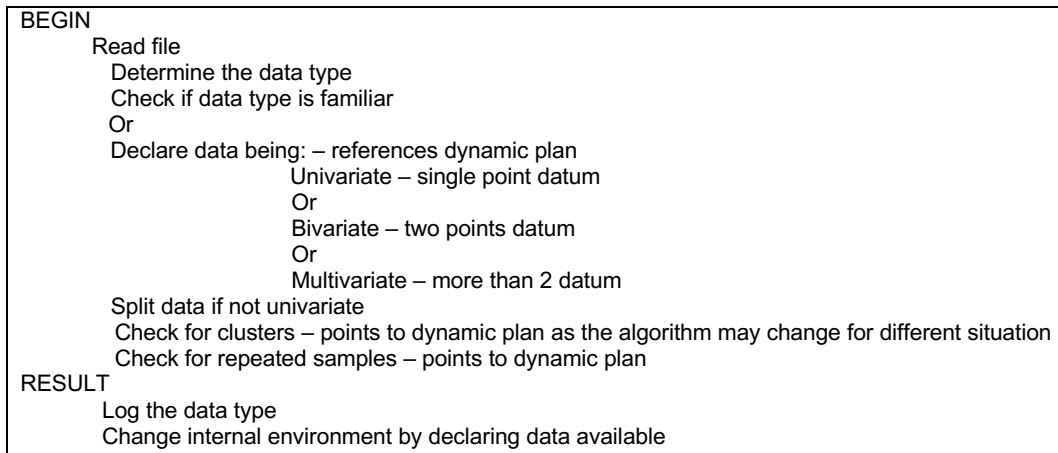


Figure 7.4: Reasoning Algorithm

In this scenario a file known to have 2 clusters with a quadratic trend and having a Gaussian variogram as a best fit by determined by a human user. The agent was intentionally left with no knowledge of quadratic type of trend it should require dynamic binding functions to be called. The file had only few data points and thus should not be split for optimization or be sampled. The file is a text file named q_gooddata.txt.

7.4.2.1 The experimental run 1

The file is placed by a human user to the folder being monitored by the VAC. The file has a sample dataset with 4 columns (ID, x, y and v). After the file was inserted in the folder the agent located and recognized it, figure 7.5.



Figure 7.5: Report of File Reception by the Agent System

Following the process for dataFinder agent, two clusters were found in this experimental data. However, only two of the clusters found were determined by the agent with its existing plan, but during this cluster finding process the agent determined that there was some data left which could be another cluster, the relation to x-y coordinate to the data. This was relayed back to the user in order to confirm that this left-over data was not a cluster, and so the agent then added this result (the inter-cluster range according to data size) as new knowledge and the plan as a new plan, in case this type of cluster is encountered at any time in the future. The agent found two clusters in the dataset, see figure 7.6 where the clusters are based on higher and lower v values. The sample cluster data that is given in figure 7.7

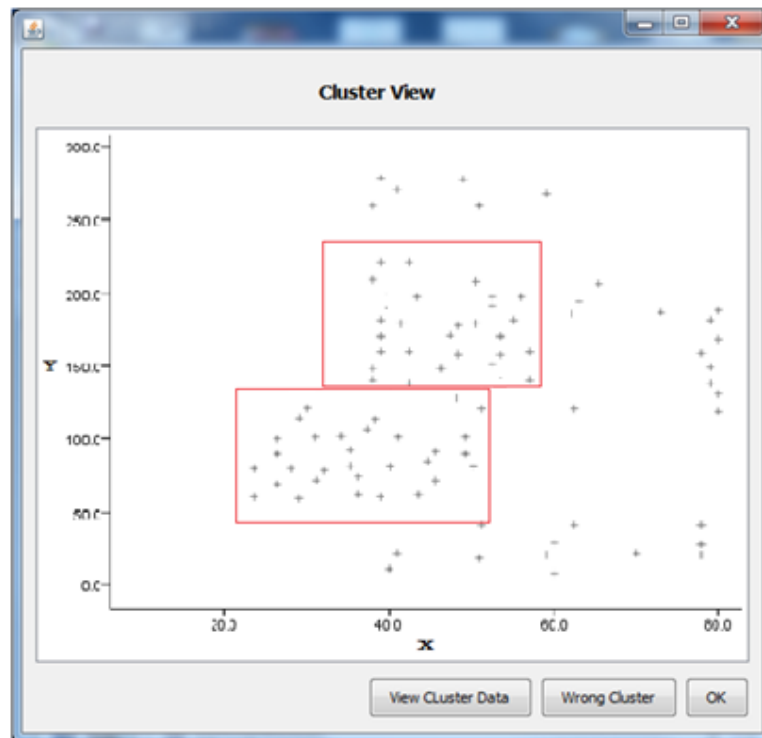
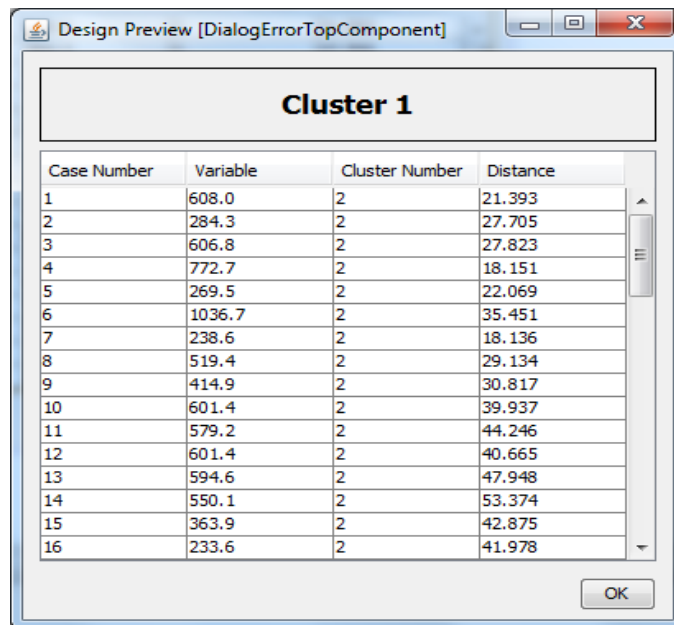


Figure 7.6: Clusters Identified by the Agent

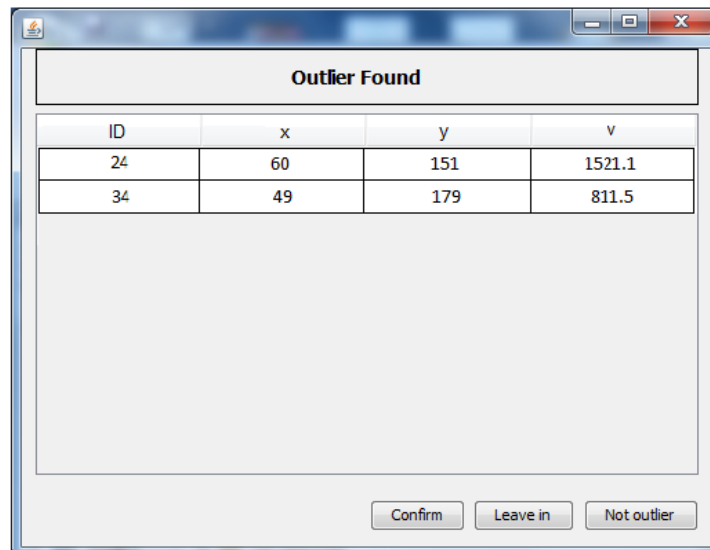


The image shows a software window titled "Design Preview [DialogErrorTopComponent]". Inside, there is a section labeled "Cluster 1" which contains a table with four columns: "Case Number", "Variable", "Cluster Number", and "Distance". The table lists 16 cases, all assigned to Cluster 2, with their respective variable values and distances. An "OK" button is located at the bottom right of the window.

Case Number	Variable	Cluster Number	Distance
1	608.0	2	21.393
2	284.3	2	27.705
3	606.8	2	27.823
4	772.7	2	18.151
5	269.5	2	22.069
6	1036.7	2	35.451
7	238.6	2	18.136
8	519.4	2	29.134
9	414.9	2	30.817
10	601.4	2	39.937
11	579.2	2	44.246
12	601.4	2	40.665
13	594.6	2	47.948
14	550.1	2	53.374
15	363.9	2	42.875
16	233.6	2	41.978

Figure 7.7: Printed File of Cluster Sample

The agent proceeded to removing outliers and the data determined to be erroneous is displayed to the user to determine if an outlier and should be removed, or not outlier and should be left in, figure 7.8.



The image shows a dialog box titled "Outlier Found". It contains a table with four columns: "ID", "x", "y", and "v". Two rows of data are shown, corresponding to cases 24 and 34. Below the table, there are three buttons: "Confirm", "Leave in", and "Not outlier".

ID	x	y	v
24	60	151	1521.1
34	49	179	811.5

Figure 7.8: Printed File Representing the Outliers Found by the Agent

The agent identifies that data split is not required and proceed. A linear trend was tested but data was found to have no linear trend, since the agent at this point only have

knowledge of linear trend. This is determined as the fit is compared before and after a linear trend was tested and found to have no difference.

However after manually working with the data on GSLib it was identified that a quadratic trend is present on the data. This new knowledge was provided to the agent via Dynamic binding mechanism for how to deal with this kind of trend, figure 7.9. Using the new knowledge a quadratic trend was tested and found to be present. The trend was handled and the fit is compared before and after the trend, see figure 7.10.

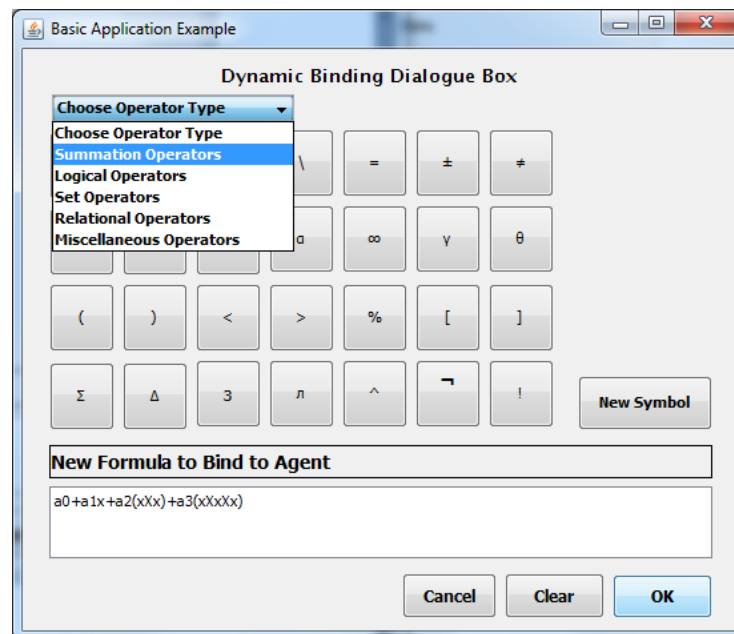


Figure 7.9: Dynamic Binding Dialogue Box (user inserts formulae for quadratic trend)

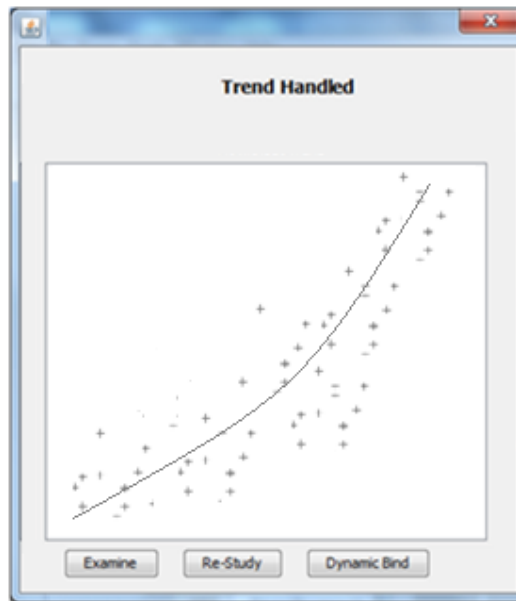


Figure 7.10: Quadratic Trend to be Handled

Since now there is more than one plan for trend analysis the agent DataAnalyser determine the process of handling trends is on its own complex and thus new child agent was reproduced, figure 7.11. This new agent has ability to handle any kind of trend (a dynamic binding function does not always cause the reproduction function to be called but when a single plan becomes complex the agent is then triggered to reproduce, such as this case. For reproduction function see section 6.4.2)

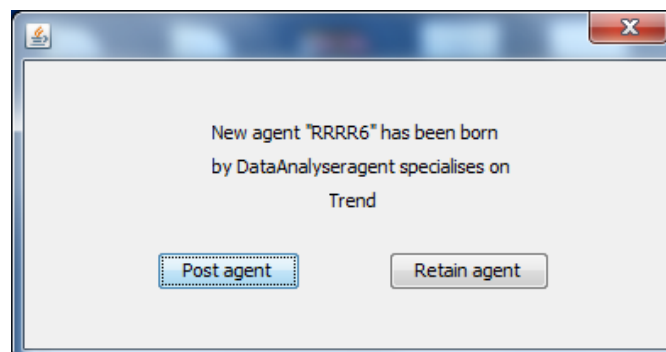


Figure 7.11: Agent Announces New Agent Reproduced

The data was cleaned and analysed an experimental variogram was produced by the agent, figure 7.12. At this stage the scale and lag size was determined by the agent and eventual choice for the theoretical variogram model was spherical. After manually constructing the variogram a human user was able to confirm the variogram to be spherical. The process of handling the data and come up with the variogram (which is the main goal of the agent system) took the human user 30 minutes as compared to the agent which took 122 seconds. This is despite that the agent stopped to have quadratic trend knowledge being dynamically bound to it. It is expected that the agent will take even shorter time to handle this type of data next time while the human user is very likely to take the same time.

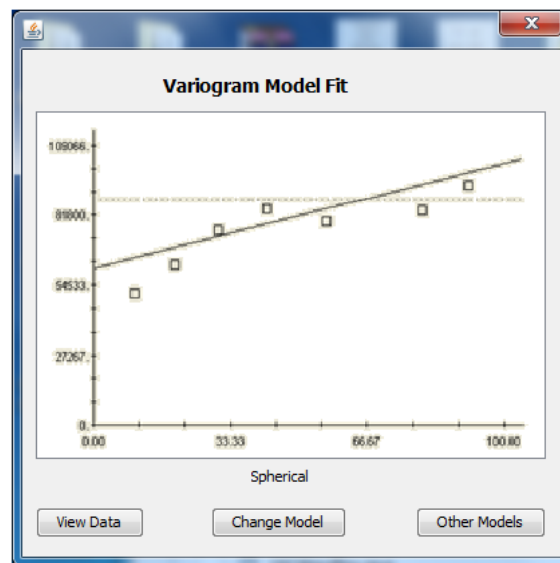


Figure 7.12: Experimental Variogram Fitted and the Derived Theoretical Model as Spherical

The agent now studies goodness of fit and in this case it kriged the data and provided a krige file which can be taken over to any tool like ArcMaps for display (Kriged maps (images) are out of the scope of this this, therefore the data file is produced and can be put into any tool for imaging).

7.4.3 Scenario 2: Testing Communication and Faulty data

Few issues were determined during the development of the communication mechanism. One of the major issues was that the agent platform used for the system development did not support the agent feature of mobility and had a very low level networking mechanism implemented to allow communication. For this reason, a structure had to be defined to allow a weak mobility-like function and a better communication function. A Broker agent was developed, which was able to segment the data if too large using known data analysis methods in GIS, like defining each cluster as a dataset and/or using stratified random mechanisms to sample the data (multi-samples to be analysed by multiple agents in this network). The mechanism that was implemented is an ad-hoc function to identify binomial descriptive data and able to divide it into multiple smaller files, for the purpose of testing.

In this scenario three files were placed at one in the folder for the agent to handle. The agent should be able to determine that the files could be handled quicker with the help of other agents. Thus, the agent should communicate with broker agent to pass the files to other agents who can handle them. The three files to be analysed by the agents have the following properties:

1. Known file, thus has been handled before by one of the agent; this should help determine if agents will identify files they have been dealt with and simply present the results instead of reworking them.
2. Corrupted file; here no agent should respond to/declare corrupt.
3. New file to all agents in the system; The file is design to have no clusters or trends simply to see that agent can collaborate and return results as required. However, this file is binomial based file. The agent currently does not know how to handle binomial data but already have a knowledge to handle each column value separately should there be more than one column (id, x and y are not considered as column value and the agent currently knowledge does not allow

it to deal with 3d data). Here this experiment is designed so to test data splitting mechanism and so dynamic binding for handling binomial data is not implemented.

7.4.3.1 The experimental run 2

The file as their placed synchronously on the folder, the agent realized of the job and flags the files being available. The agent send a message via the broker agent (to all agents and human user, figure 7.13) that there is few files it would like analysed by the help of other agents in the system. The agent keeps the dialogue for 5 seconds then continues by announcing for help to other agents while simultaneously disable the 'Announce to Agents' button.

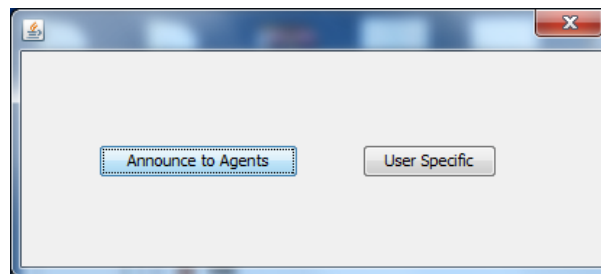
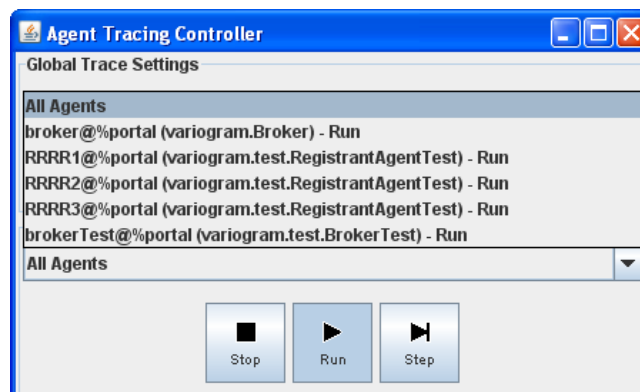


Figure 7.13: Announcement to Human User (if no response the agent continues)

The JACK trace function (shown in Figure 7.14.) shows agents RRRR1, RRRR2 and RRRR3 have responded to the call. The actual Broker agents utilise a copy of itself called BroakeTest to act as the lookup table while the other acts as a searcher for available agents in the network, respectively.



7.14: The Trace Function of VAC, Communication

The broker agent shows 3 agents have responded to the call hence all 3 file have been taken up by other agents and thus the originator agent is left to only analysed the result provided by the responded agents. The agent analyse the result using algorithm provided on section 6.5. As the process continues; firstly the originating agent received results of the file from agent RRRR1 (the same agent as in experiment 1) that it has dealt with the kind of file before and the resulted Variogram was Spherical and shown the same figure as figure 7.12. next the agent RRRR3 return results and determines the file being corrupted, figure 7.15.



Figure 7.15: Message Show an Agent Announcing a Corrupted File

For confirmation Broker agent resend the file over the network for verification and the agent RRRR0 takes it and also return corrupted file message, figure 7.16. it is important to note at this point if agent RRRR3 did not send the corrupted file back to the network to be re-checked human user still could opt to manually examine the file through the 'examine' function on the dialog.



7.16: Faulty File Announced and Both Human and Agent Users Have the Possibility to Re-examine

The file handled by the agent RRRR2 has binomial data and can be split into smaller file for handling efficiency, therefore more complex process. The process starts with agent splitting the file where each column (two unrelated chemical on the x-y coordinates) is examined by its own. Hence, creating 2 new files and place a call to broker agent to call for help to handle the files. The broker agent places them to the examination folder to be handled and the process restarts, figure 7.17. The process is delicate since the broker agent is already engaged by the originating agent (RRRR0) and therefore it has to keep integrity and be able to determine which agent process is handle by which agent. Especially that agent RRRR0 is considered to be like any other agent in the network and could take up the task called by RRRR2 (which happen to be its own sub-task).

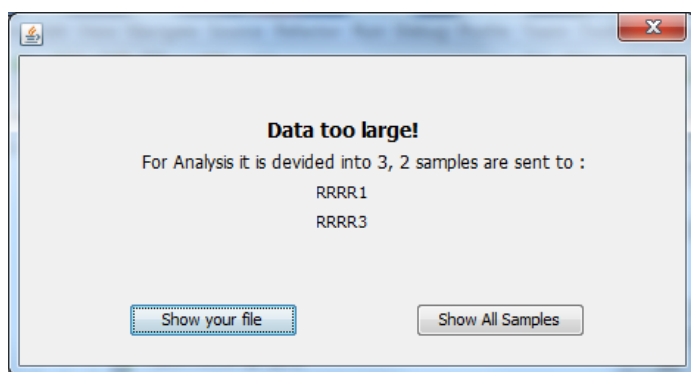
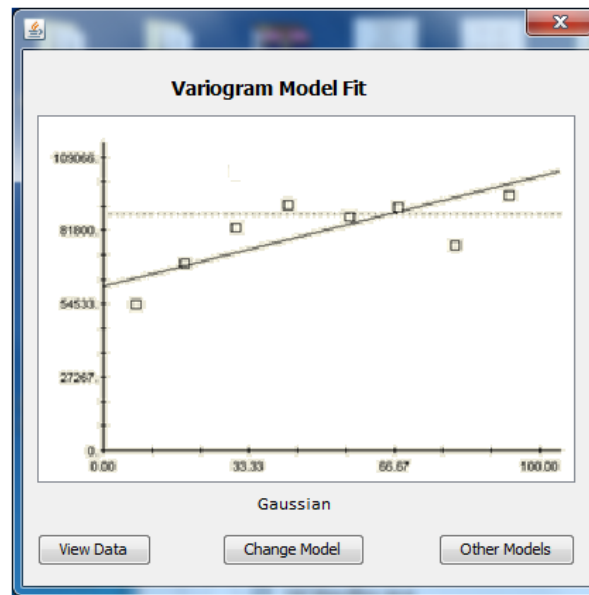


Figure 7.17: Agent Receive Large File and Splits it Then Place New Files to The Examination Folder

Agent RRRR1 (has finished its previous call) accepts the call from agent RRRR2 to take one of the 2 files, analyse the file and finds no cluster and no trend present on the file and produce a variogram shown in figure 7.18.



7.18: Experimental Variogram Fitted and Agent Derived Theoretical Model as Gaussian

Using the same process of section 7.3.3.1 Agent RRRR3 finds one cluster and a trend being present on the file (figure 7.19, and figure 7.20 respectively) and produces a variogram shown in figure 7.21.

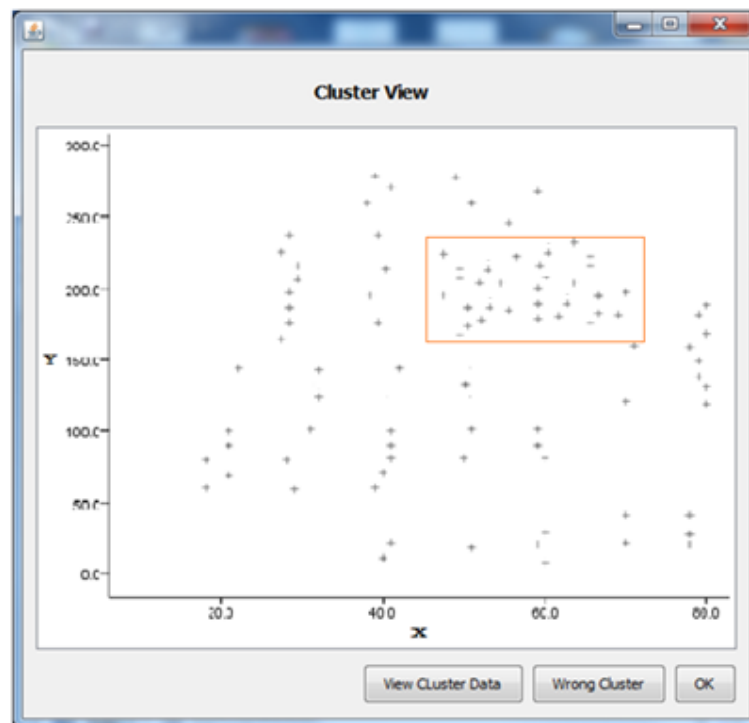


Figure 7.19: Cluster Identified by the Agent

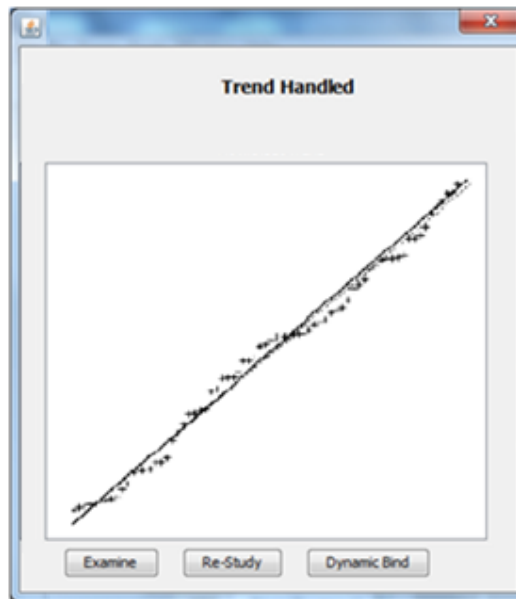


Figure 7.20: Linear Trend to be Handled

Agent RRRR3 found the data to have a Gaussian variogram, figure 7.21.

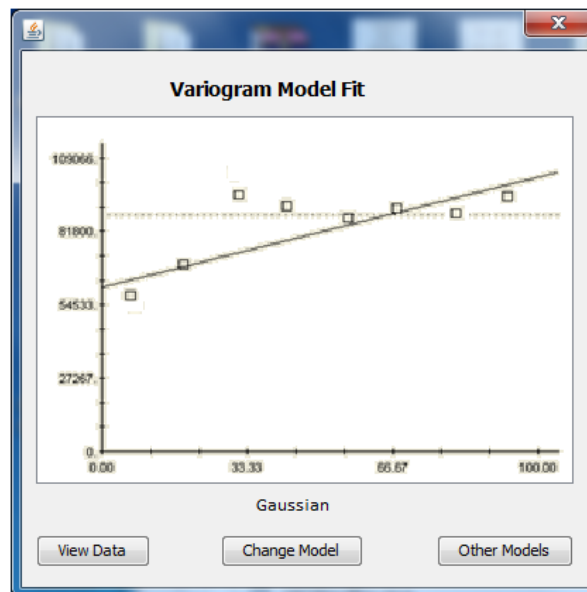


Figure 7.21: Experimental Variogram Fitted and the Agent Derived Theoretical Model as Gaussian

The process of fitting the variogram from initial call by agent RRR0 has taken 452 seconds (7minutes 32 seconds) where the replicated process done by human user took 3 hours 2 minutes and 32 seconds. It is clear the agent provide better functionality in terms of resources management and data processing efficiently, which allow a rapid

development of variogram. This has major implication on areas that requires real time information of spatial dependency.

7.5. The Agent handling the Walker Lake data

In this section, clusters and trends are detected then highlighted and discussed, see Figure 7.22.

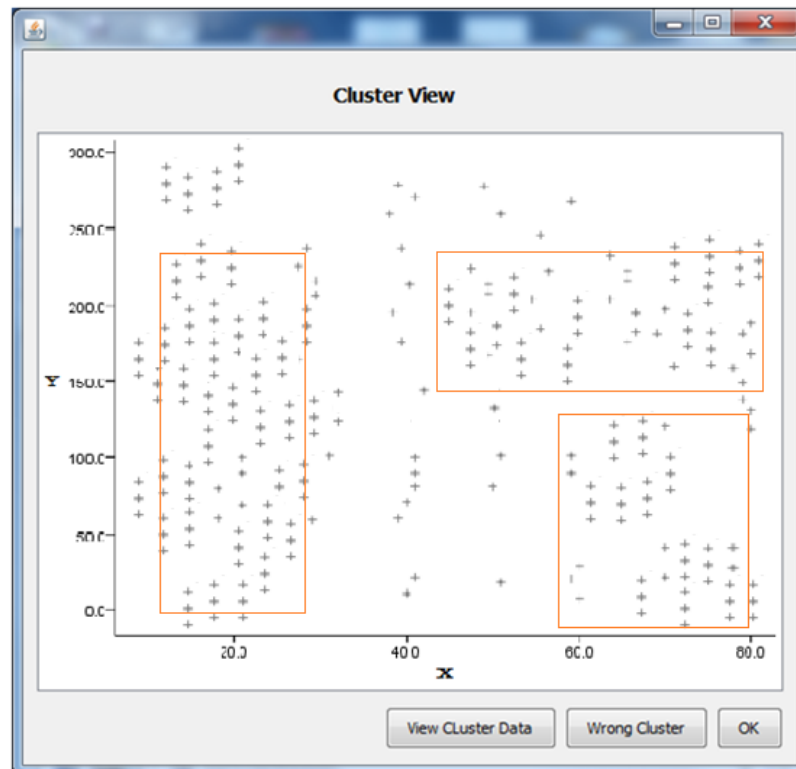


Figure 7.22: Identified Clusters

Isaaks and Srivastava (1989) data have indicated that there was more than one cluster of data, which was confirmed by the agent as three clusters were found (see Figure 7.22). Each set of data was realised to have some clusters, while other datasets did not (these were denoted as having one or two clusters as indicated on the experiments scenarios in section 7.3.1). The intelligence and learning mechanism was demonstrated at the cluster analysis and separating clusters stage, when the agent determined the structure of the data to be analysed, which can be aided by the Data Analyser agent. Then a sample iteration factor was input and checked to see if this sample was

appropriate (this could be achieved by the human expert interaction, other agent interaction or the agent self-confidence mechanism provided). The self-confidence mechanism was achieved by looking at two important aspects:

1. the number of data which appear in the cluster compared to the total number. If the number of data in the cluster is less than 20% of the overall data, then remove the cluster data and check if the left-over data provide a normal or random distribution with no clusters. If so, then the initial analysis was correct. Otherwise, either there were no clusters or there was more than one cluster. In this case, increase the number of iterations.
2. checking the separation of the identified clusters compared to the mean separation. If this value is lower than the mean of the overall data, then these two or more clusters could be perceived as one cluster, but due to too many iterations they were classified as one and thus the next time reduced the number of iterations.

The expert user or other collaborating agent can help with learning and making intelligent decisions via the GUI provided by this agent. The GUI shows the data on a scatter plot and draws contours around the perceived clusters, while numerically identifying why it chose these data points as clustered. The preferred results and the undesired result get added into the agent's 'brain' and will be used the next time data arrives and is perceived to be similar to this data.

The agent's current knowledge allows it only to determine clusters and/or a linear trend in the data, if there is any. Using this mechanism, it determined that the data had three clustered areas (as shown in Figure 7.22) and found outliers. It also found that the data had a trend which was not analysed by Isaaks and Srivastava (1989), which the agent dealt with when it produced what it determined was a better fit for the variogram. The trend identified was found to be linear in a North West direction at a 70 degree angle.

This trend analysis was achieved using the polynomial regression formula presented previously in section 4.3.3. To analyse this dataset, a clustered and stratified random technique was chosen by the Sampler agent when sampling the data.

Comparing these results with the original results determined by Isaak and Srivastava (1989), the agent identified a few minor differences. For example, the original authors only found two clusters and did not find any trends. The agent considered the relation between the U and V values and their relation to each other where Isaaks and Srivastava (1989) only consider the individual variable and their relation to x-y coordinate. Other analysis like the outliers had the same outcome to those found by the VAC.

The agent determined that the data had three clustered areas and found outliers similar to those found by Isaaks and Srivastava (1989). The data was found to have a trend, which was dealt with before producing a variogram. Clustered and stratified random techniques were chosen by the agent when sampling the data. Results from the sampled data were compared to the whole data. The results of checking for clusters, as presented by the agent platform, can be seen in Figure 7.23. There were also other analytical differences in results compared to Issak and Srivastava (1989). For example, only two clusters were found by the authors but three were found by the agent, and the agent also found a trend which the authors did not examine at all.

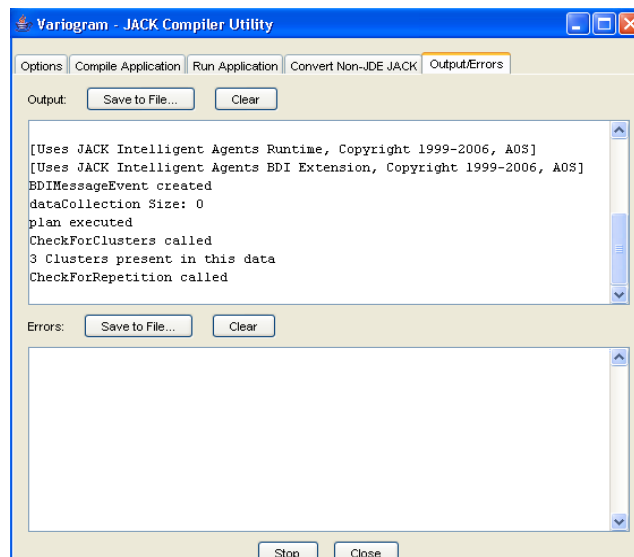


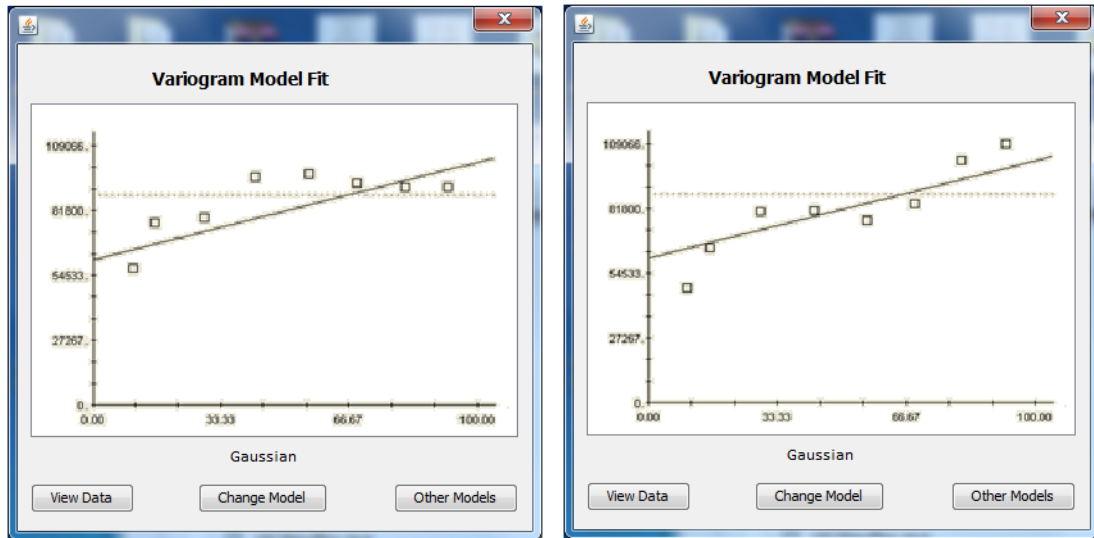
Figure 7.23: Clusters Identified in the Walker Lake Data

The results produced by the agents are slightly different to those provided by Isaaks and Srivastava (1989). However, the differences should be acceptable since the theoretical model which was fitted to the variogram was the same. The agent provided satisfactory results with regard to the GIS methods concerned, particularly when it is considered that, in GIS, analysis of the same data does not always provide the same results - different experts can provide different results, depending upon which strategy they use (Issak and Srivastava, 1989; Ripley, 1981; Cressie, 1993; Deutsch and Journel, 1998).

7.5.1 The Variogram Selection Process

Once the data had been analysed and cleaned, the variogram was constructed using defined models. All the lags of these models were presented to the user and a selected best fit model highlighted by the agent. Here, for the purpose of variogram construction, all parameters were chosen and then inserted into a standalone package that could do the drawings. These drawing were developed to be undertaken by the Plotter agent, but due to time constraints and the fact that with these drawings it makes no difference as to where and which program draws them, the important values were defined and passed to an established graph drawing package. These graphs are presented below, followed

by the fitted variogram. The corresponding variables (lags, sills and ranges) for variograms being chosen are presented in Appendix D.

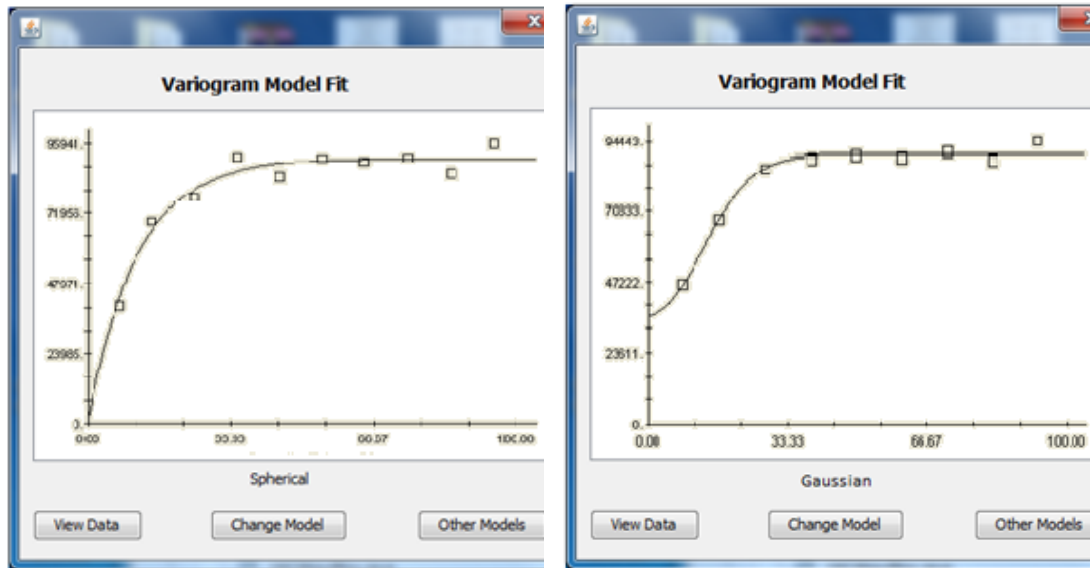


7.24a: Anisotropic Experimental Variogram at 45 degree angle

7.24b: Anisotropic Experimental Variogram at 90 degree angle

Figure 7.24: Experimental Anisotropic Variograms

After determining the various experimental variogram curves, the fitting process is started by the ModelFitter agent. In the Walker Lake dataset, two curves were determined to be most suited, the spherical and the Gaussian. In the first iteration, the spherical variogram was found appropriate, which was the same as chosen by Isaaks and Srivastava (1989). Figure 7.25a shows the variogram determined by the agent and which is the same as the variogram drawn by Isaaks and Srivastava (1989). However, after analysing the data further, the agent also found a linear trend in the data and decided to remove it. Analysis after the trend had been removed led to the decision for a Gaussian variogram instead of a spherical one (contradicting the authors), as the agent determined that this was a better fit (see Figure 7.25b).



7.25a: A Spherical isotropic variogram determined by agent prior to remove trend

7.25b: A Gaussian isotropic variogram determined by agent after to remove trend

Figure 7.25: Fitted Isotropic Variograms

This best fit was analysed using various well established geostatistical tools for variogram validation, determining the goodness of fit, using ordinary kriging and cross validation (see Chapter Four).

7.5.2 The Variogram Validation

The goodness of fit of this variogram was assessed using ordinary kriging and cross validation. Here again, the numerical data results that determined the acceptance of the Gaussian variogram were put into the GSLib and ArcGIS applications. The results are presented in Figures 7.26a and compared to the Isaaks and Srivastava 1989, figure 7.26b. The agent's suggestion on the final result was Gaussian as opposed to the original authors' of spherical distribution, which is what was determined by the experts (original authors) for the given data.

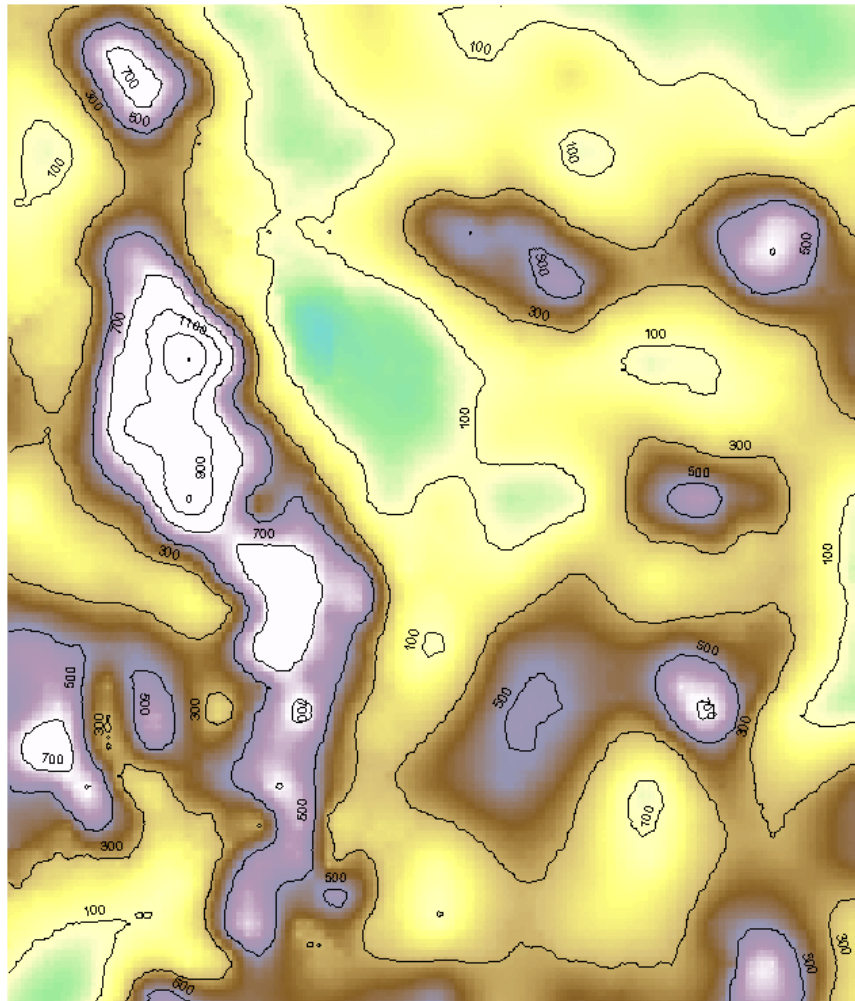


Figure 7.26a: Kriging Map of Walker Lake Data as Identified by the VAC

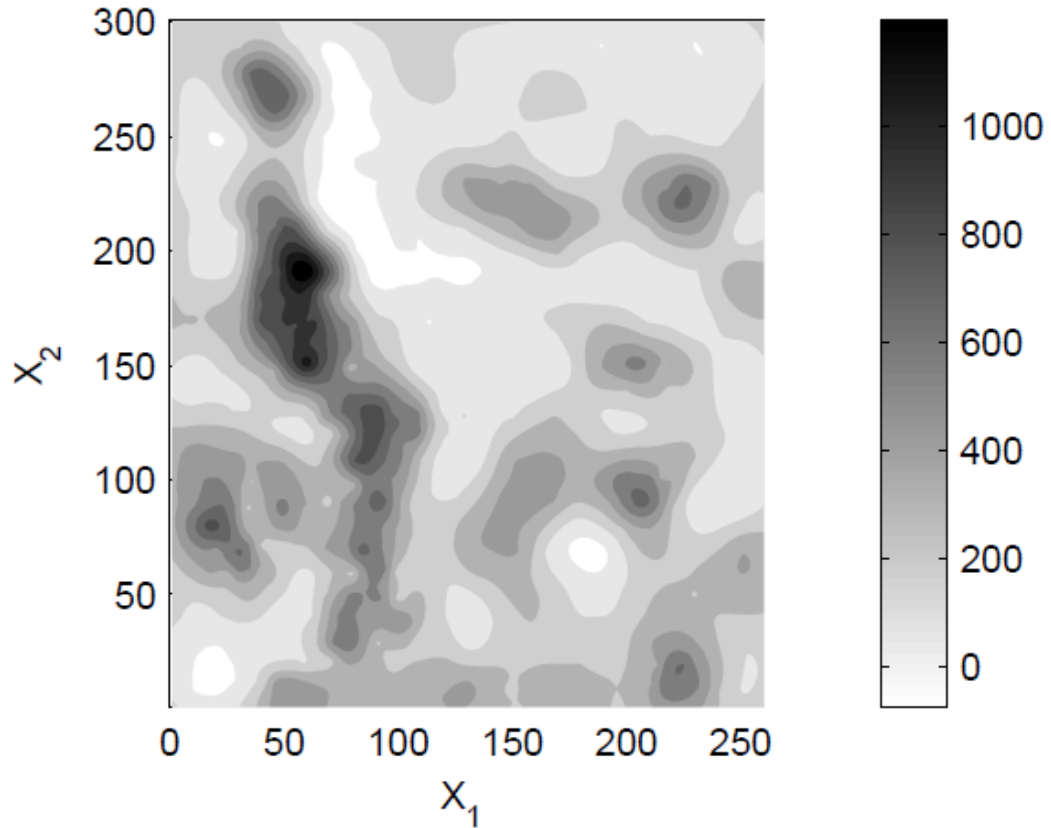


Figure 7.26b: Kriging Map of Walker Lake Data (Isaaks and Srivastava, 1989)

The kriging process was undertaken in two dimensions with the interpolation interval (x - y) calculated at 2.8588 and 2.8020 respectively. This kriging process was performed on variable V of the Walker Lake dataset, but renamed as z . The process is done starting at (x - y , 8-8) with a range starting from 8 to 251 for x and 291 for y , giving a z estimate at point 0-0 over the range of -23.47 to 1314.25, giving a z estimated standard deviation range of 179.71 to 292.42 with a mean value of z estimated at 298.6551 and its standard deviation mean at 38884.30. The residual errors based on a held-back sample and some additional points taken at random from the exhaustive data, as provided by the agent, were input into a map and are presented in Figure 7.27a which was also proven to be very similar figure 6.9 in Isaaks and Srivastava (1989), see figure 7.27b.

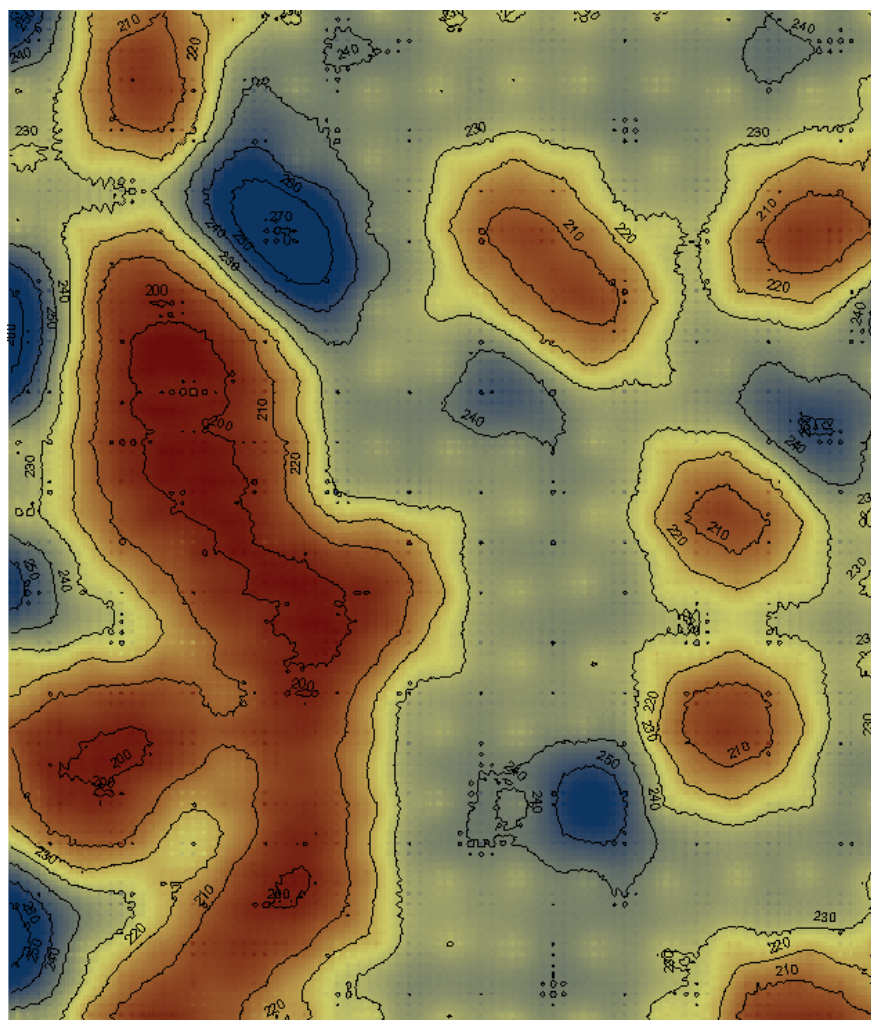


Figure 7.27a: Map of Walker Lake Data Showing Residual Errors From the Kriged Data by VAC

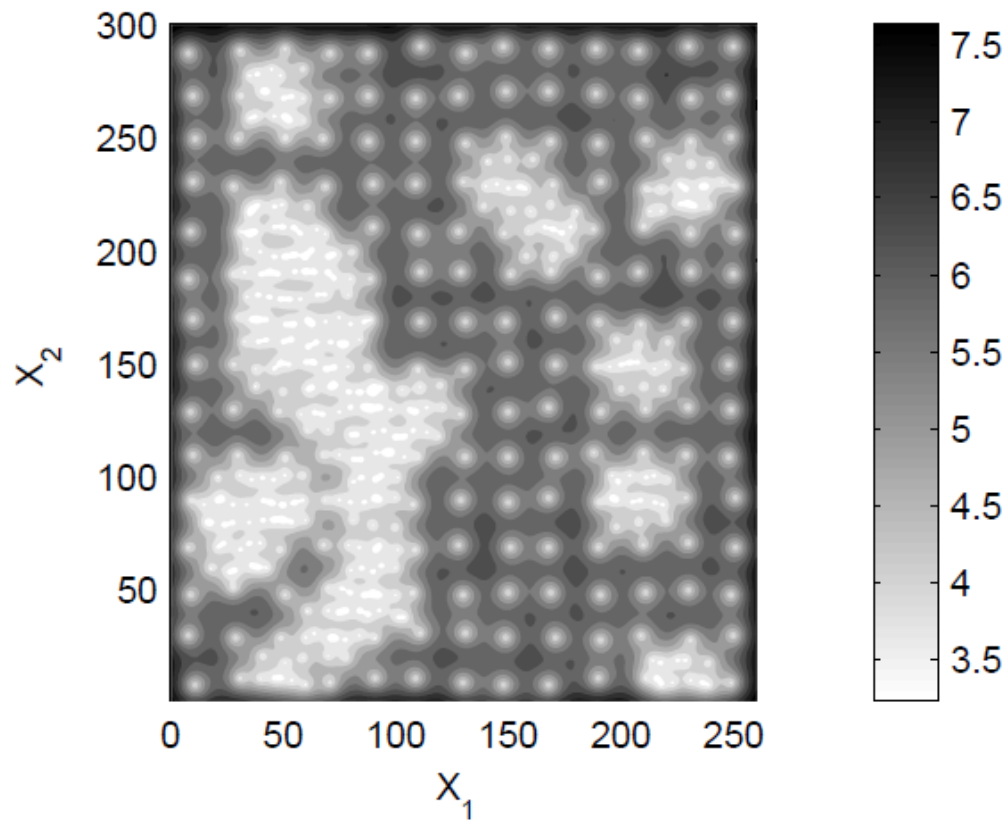


Figure 7.27b: Map of Walker Lake Data Showing Residual Errors From the Kriged Data Isaaks and Srivastava (1989)

The cross validation graphs show that the agent underestimated some of the higher points and overestimated some of the lower elevations. This is because the values of the Walker Lake dataset consist of points taken over an area containing a ridge and a basin. The points taken on the ridge are shown on the kriging map (Figure 7.26a) as being darker brown, while the points on the basin are shown by the lighter coloured area. These points have caused the regression line to tilt slightly off the 45 degree. Since there was a linear trend found in the data, the agent decided to remove this to try and obtain a better fit. This improved the cross validation at about a 47 degree angle with 95% confidence intervals. Figure 7.27b shows the residual error of the kriging process, with darker red being the highest and darkest blue being the lowest. This was determined against the held back sub-sample of 144 random points, held back from the original 470 sample points of the Walker Lake dataset.

As most of the process of reaching a chosen good fit variogram requires the nearest neighbour values, the agent was instructed to define a file with these values. This file was then tested to see if the nearest neighbours chosen by the agent corresponded to other available tools. Twenty random values were selected to determine their nearest neighbour. Each value and its corresponding nearest neighbour were collected together with distance and difference. These values were then saved on a special file, so they could be used for finding trends and other functions.

7.6 Comparing the VAC to Other ‘Service’ Based Agent Technologies in GIS

The system was compared to two well-known agent-based GIS - the RePast and Oracle spatial. The results show there are significant deficiencies in these two systems, addressed in terms of performance, functionality and complexity of execution. Also, the most important feature in the system proposed is the possibility of having a GIS tool that can talk back to the expert and co-expert analysing a spatial phenomenon. This test will only use one agent from the VAC. This agent will be the Data Analyser agent, as its functionality is very similar to that supported by RePast and Oracle.

The test will be to measure the run of each system in order to produce the results. This will be a measure of speed (in seconds) where the fastest is the best. This will lead to an examination of the agents’ functionality in each system and the query complexity (defined as length of writing and number of nested loops required where these features will take up more computing resources). An agent’s functionality will be assessed by the amount of functions an agent can provide. The functionality measurement will comprise of the amount of features the agent possesses (the agency), the data acquisition process and the querying process. The query complexity will also be looked at, based on the amount of knowledge required from the expert to construct the query. This test is not intended to accurately examine the result, as in GIS different analyses can provide different outcomes. For this reason, we have refrained from placing any emphasis on

the accuracy of the results and simply compare them to those provided by the expert user.

In GIS, this agent framework and architecture offers flexibility to the user. It offers the capacity of easy extensions, as agents are able to move from one system to another if more resources are required. Also, the agent is learning from the user, so that in time the agent becomes an expert and requires less and less intervention from the user to solve problems. Given the range of data types and contexts in which the agent system might be applied to, this is the main advantage of applying agents as processes and service providers, rather than static objects (simulating entities under study) like most current GIS tools which use agent technology.

There were also some functions that were required from the agent's point of view. These functions were not theoretically novel, but were not implemented due to the current structure of agent programming languages. This issue was addressed and a structure was proposed and tested. The structure, termed dynamic binding mechanism, was where an agent could suggest a solution and negotiate with a human or agent expert. If the expert manages to convince the agent that their technique is better, then the agent will request the expert to feed this technique (as a formula) to the agent, so that the agent can then add this information to its knowledge domain.

After the agent's mechanism was enabled in Oracle, a select SQL was used to attempt to minimally analyse the data. Thus, the measurement of the time taken and the results of the query were recorded.

Table 7.1: Comparison of Agents Run

	Seconds	Measured on a scale of 10		
Technology	Run to completion	Agent functionality	Accuracy of the results	Complexity of query structure
Oracle	3.1	3	3	10
Repast (ArcGIS)	6.4	3	5	4
VAC	2.3	6	7	1

In the experiment, we were not able to utilise the agent to provide any help in data analysis, but we were able to utilise the agent for distributing the data. After the query was executed, the agent was able to send the results for different node (agents) execution. This allows parallel execution when needed, and indeed help on the speed of the execution run (the analysis). In VAC, when the agent is in running mode (executing), the dataflow is not only being handled internally but is also being communicated to the human expert. The human expert's view of this data is shown in Figure 7.29.

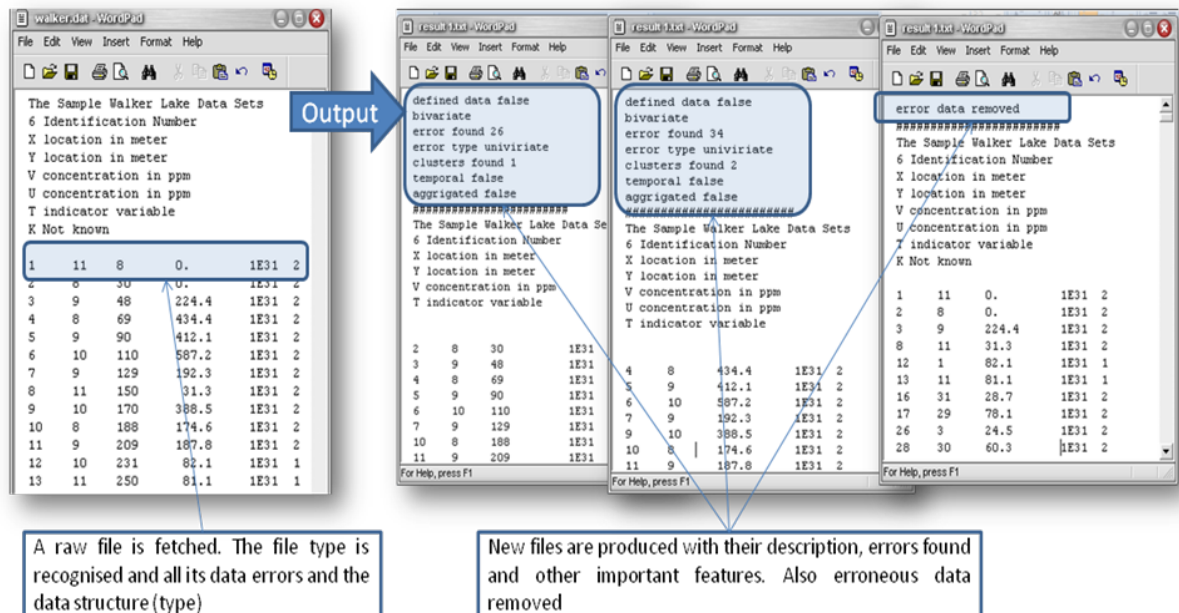


Figure 7.29: Exhibition of the Data Analyser Agent

The data was found to be bivariate, with each variable having clusters. So it was divided into univariate and clustered data.

On the other hand, the data was analysed using the RePast of ArcGIS. This examination was able to examine the errors by simulating the environment. This is the only feature provided by RePast. However, it provided the possibility of seeing the errors. Even though ArcGIS have improve the functionality of RePast by developing a more comprehensive tool (Geostatistical Analst, 2009), the tool uses very minimum agency functionality and in this particular example it function exactly as RePast.

For Oracle, after the agent mechanism was enabled, a select SQL was used to attempt to minimally analyse the data. Thus, the measurement of the time taken and the results of the query were recorded. The results are shown in Figure 7.30a. The same data analysis was carried out using RePast on ArcGIS. It was only possible to examine the errors by simulating the environment. This is the only feature provided by RePast. However, it provided the possibility of seeing the errors. The results are shown in Figure 7.30b.

```
SQL>
SQL> EXECUTE WALTER_FUNC
```

Output--

NAME	X	Y	V	T
1	11	0.	1E31	2
2	8	0.	1E31	2
3	9	224.4	1E31	2
8	11	31.3	1E31	2
12	10	82.1	1E31	1
13	11	81.1	1E31	1
17	29	78.1	1E31	2
28	30	60.3	1E31	2

7.30a: Oracle Query

RePast Output

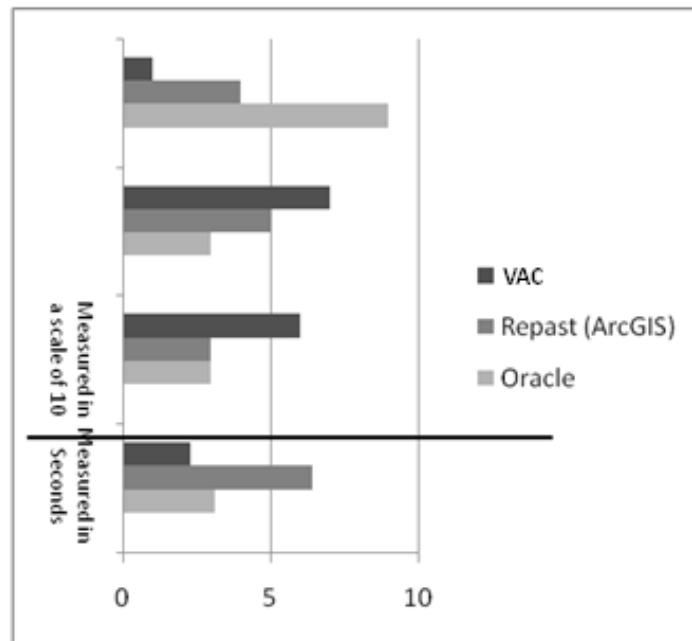
1	=	1
2	=	2
3	=	3
4	=	4
5	=	5
6	=	6
7	=	7
8	=	8
9	=	9
10	=	10
11	=	11
12	=	12
13	=	13
14	=	14
15	=	15
16	=	16
17	=	17
18	=	18
19	=	19
20	=	20
21	=	21
22	=	22
23	=	23
24	=	24
25	=	25
26	=	26
27	=	27
28	=	28
29	=	29
30	=	30

7.30b: RePast through ArcGIS

Figure 7.30: Results From Oracle and RePast

This test produced satisfactory results, given the fact that one geographical problem can be tackled in different ways. Figure 7.31 shows the summary of the results from the

experiment. From the perspective of developing agent-based systems for GIS, this test produced satisfactory results, particularly given the fact that any one geographical problem can be tackled in a number of different ways. Each expert tends to take their own approach to analysis, although hopefully converging on the same outcomes and decisions.



Note: x-axis represent a ranking scale of 0 - 10

Figure 7.31: Comparison of Agent Performance

The overall results of the investigation are given in Table 7.1. The results show that it was possible to achieve similar results to those produced by the Data Analyser agent. However, this involves constant interaction with the expert user. It also adds its own complexity, as the expert user needs to also be a SQL query designer, as the query used here is complex PL/SQL with a number of 'nested loops'. The query process took 3.1 seconds, which is a less than a second slower than the Data Analyser agent. This difference is negligible, except that it took over an hour to prepare the query for the Oracle, while the Data Analyser agent requires no query at all. The conclusion is that the agents in Oracle are useful for a distributed data environment, particularly in

heterogeneous operating systems, but they are not facilitated to deal with intelligence, mobility or any other software agent characteristics.

7.7 Conclusion

This chapter provided the results achieved by the VAC by experimenting on its functionality and comparing its results with established authors. Through these experimental runs of the VAC, a few problems were identified. These problems were categorised into their functionality, the effect on the outcome or the system being internally and externally validated respectively. These problems were solved by either being rectified or theoretically justified to pose very little hindrance to the performance of the system.

The main aim of this research was to provide an agent-based system that would provide a service as an expert in a distributed environment. The functionality is to leverage this in utilising available resources. The VAC was then compared to other agent systems that have been published as service providers in GIS and found to be very different. Thus, the system is a new one of its kind and is useful as a system that could be implemented into GIS simulation tools. Apart from having the VAC as an original contribution in GIS, one other contribution to agent development is the dynamic binding mechanism which allows plans to be added to the agent during the run time.

The next chapter will cover the determination of the tests and functionality of the five characteristics of software agent previously discussed, in order to be able to enhance GIS functionality, and improve and encourage utilising the computing capacity. Furthermore the limitations, advantages and disadvantages of the VAC will be discussed.

CHAPTER EIGHT: CONCLUSION AND FUTURE RESEARCH

8.1 Introduction

Through time, geostatisticians have come to realise that by borrowing from other technologies, GIS and in turn geostatistics could significantly enhance their functionality. One reason to incorporate a more versatile computer technology is due to the growth and the move towards ubiquitous, wearable and mobile computing (Brimicombe, 2003; Poslad, 2009). As computers keep following the Moore's Law of doubling the capacity, while the price decreases and make them available in most homes in western world and accessible almost anywhere in the world (geographical space). This expansion causes degradation to data integrity while communication channel become more constraint due to large volumes of data (Ding, 2007), it is essential to find the means to optimise the available capacity especially on processes that require large computing resources. Geostatistical research often needs its own unique process to acquire the data information (Diggle, 1983; Isaaks and Srivastava, 1989; Cressie, 1993). These are the bases that propelled this thesis to examine the functionalities of agents in distributed component GIS, using geostatistical tools as they provide complexity challenges.

This chapter looks into the operational functionality of the tool developed for these experiments, the VAC. It will focus on intelligence and other agency characteristics, outline the issues of GIS that it intends to solve, the contribution of knowledge to current GIS systems and science in general, then provide concluding remarks and ideas on required future work.

8.2 Operational Functionality of the VAC

Building agents for GIS proved to be a complicated and complex task. The existing agent platforms had limitations on making dynamic Agent based GIS solution. Therefore new algorithms on the agency were developed and especially new agent design component was added to the JACK platform.

The limitation on the actual agent development platform can be seen on chapter 7, internal validation of the system where agents halted their process various times due to two different errors. The errors were none-response where an agent stop functioning and causing the system to stop responding to any activity and the system crashing due to large dataset. Solution to these problems had to be hardcoded and be developed through user defined API. The agent design platforms reviewed were missing important functions essential for making component based GIS system as envisaged by this research. For this reason a new agent development structure had to be established and then added into TROPOS development language as one of its agents design stages. These diagrammatic presented processes allow for easy understanding and deployment of the agents, see figure 6.19. Also precisely maps the functional requirements of each agent, their interaction and their stable status within the system acting like *class diagrams* on Object Oriented Programming. These were then published to software development community for verification, see appendix G for published material from this thesis.

Thereafter a deployment (programming) platform was established according to its mechanism, ease of development for complex system and the agency functionality. Since the VAC had to be intelligent, reactive and proactive JACK platform was chosen. However, JACK was also found to have limitations with mobility and communication functions. These functions are important for brokerage of the GIS components. This was

overcome by developing a broker agent with hardcoded solution that allowed for a cross platform communication.

The research was to assess the effectiveness of agents on the existing standard platforms (e.g. JACK) to improve and ease GIS issues pertaining to interoperability and heavy process. After examining various characteristics it was deemed that for solving GIS problems agent platforms have to be robust on cloning and communication functions. In terms of cloning agents have to be more constructive since most GIS problems are related to standardisation, mechanistic and data heterogeneity.

Even though Ali and Moulin (2005) showed their model was able to facilitate a solution to these problems there remain an issue of constant monitoring and re-development of the agents. To alleviate these problems two new features were introduced: the reproduction feature that is developed through a technique for dynamically binding new processes to already running agents and communication that allowed better data and function brokerage across platforms. These features were thoroughly described and tested on chapter 6 and 7.

8.3 Intelligence Focus in the VAC

To gain intelligence, the agent is able to log the geostatistical tool used for the decision it made. Initially, the logs would be reported to a human expert, so that the expert could point out mistakes and criticise the decision. If it was not good enough, then the agent would repeat the analysis or the expert would simply suggest a better decision and then the agent would work the decision in reverse so it could try to capture at what stage it made the mistake and rectify the log at that point. This mechanism expands the knowledge of the agent more rapidly and allows a much better relationship between the human expert and the agent thinking mechanism.

Other methods that are used to rectify a decision are achieved by the Variogram agent checking on itself. This would be achieved by using its previous logs and if the result is expected to trigger a certain action (in this case, it could be producing a certain graph or table, or even just a sentence stating a decision) and this action does not take place, then the agent could reverse work from the action point of view so that the point that caused the difference could be noted into the log, as shown in Chapter 7 section 7.4.2.2. At this point, it is important that the agent can receive feedback on its actions, so that it can decide what was wrong with the decision (if the decision was wrong) or whether it was just that it was a different phenomenon with similar data characteristics to another previous process. This will facilitate the purpose of fine-tuning the agent's knowledge.

This learning mechanism can relate the relationship between the support and distribution of the data for a spatial process on the structure, as presented by Atkinson and Tate (2000) who explain that it is important to evaluate the support of the sample data and the support of the intended final estimate. If these two are different, then one should deploy the Negotiation agent so the support for the process at large can be derived. This structure is required for the many problems that exist with the estimation tools (e.g. the tool used on the process was such that the mean is kept constant and only the variance is changed) (Isaaks and Srivastava, 1989). This is due to the fact that, in this situation, it is arguably better to do some form of correction than not do any at all. The point to stress here is that, in GIS, a data analysis can yield a number of different results depending upon the analysis structure and statistical formulae, and thus one just good enough result could produce good results elsewhere (Isaaks and Srivastava, 1989). So, as long as the results are deemed 'good enough', the agent system is said to be sufficient. In terms of intelligence, the agent has a back propagation mechanism. In context, if the machine is to think like a human, having intelligence, it is important that it also behaves like a human. Human behaviour is needed to observe a decision and, if the result is not satisfactory, rethink the decision by adding the current decision as a weight (negative) to what it already knows, so it makes a better attempt at formulating the second decision.

8.4 Suitability of Agent Characteristics to GIS

The VAC system is supported with agent characteristics that involve intelligence, mobility, autonomicity and reactivity, together with the ease of extensibility. It was also determined that proactivity is a requirement of such a system.

The intelligence of the VAC is argued to have been attained through clever agent design and the usage of the chosen agent programming platform. JACK is an environment which supports BDI through a hard coded capability, goal, plan and event alignment mechanism. Furthermore, the intelligence is further strengthened through the learnability of the agent. This is achieved by keeping the data structure in the memory bank, which reduces the process the next time it encounters the same kind of data, or a dataset which has a similar structure.

To improve learnability, a new feature is introduced through this thesis (the dynamic binding mechanism), which is currently not available on other agent systems, as it was not found within any available literature or agent systems that were reviewed. This function is derived from the argument that, in certain circumstances, agent cloning is not enough and a reproduction mechanism is required. Through such a reproduction mechanism, the dynamic binding function of the agent was introduced. This function is close to reproduction, as it changes the physical structure of the agent and its previous capability, so it is suggested (even though not implemented in this thesis) that using agent cloning and then dynamically binding the cloned agent will introduce a new agent very similar to the original agent but not identical - this is what is termed 'reproduction'.

For mobility, a brokerage mechanism was introduced. Many algorithms were suggested for this function, and some major thinking was required as JACK does not support mobility (this was confirmed by the JACK development team). Thus, functions using the

communications that are available within the JACK environment were adopted, with some modification (together with the introduction of the Broker agent). In addition, through this adaptation of the communication mechanism, the design of the Broker agent allowed social ability characteristics to be achieved, by having the agents report their functionality to one another via the brokerTest agent, which is the agent responsible for the lookup table that controls agent function over their platform.

Since the VAC makes periodic checks into data repository areas (a pre-determined location from which the dataFinder agent can fetch data), it uses software agent autonomicity characteristics to take initiatives to respond to file (dataset) changes and react to them. This achieves the requirement of agent characteristics of reactivity, which is also supported through an inter-agent response mechanism where each agent will react to certain events, a feature provided and pre-implemented by JACK. The reactive feature was experimented on through internal validation and at some point was determined to cause a problem (tested on the IntegrityChecker agent of the VAC). The problem was that the agents would not react to a dataArrive event. This problem was found to be caused by having a data (file name) error or not being available at the perceived location. The problem was eventually solved by introducing a dedicated location for data fetching and emphasising the data file name to be correct.

The proactive characteristic of agents was also determined to be important for GIS, but it is not implemented through this thesis as it was determined to only become important when the system grows very large (having agents over a Wide Area Network) and needs to work in multiple sub-networks, where, when idle, the agents are expected to take their own initiative to proactively deal with datasets and keep them ready for information (if required) by other agents. The current system was only tested over two networks (Local Area Networks) which are directly connected and do not reside in their own (Wide Area Network) environment. Thus, this characteristic was deemed important and kept as being a future improvement of the VAC system.

Another function that the VAC achieved was ease of extensibility. This function is especially required in fields like GIS, where analysis can be done differently to achieve the same results. This function is derived from the ideas of Anselin (1995 and 2005) and Wiederhold (1994) on ontologies in GIS. As the intelligent agent provides ontology, the described structure could work by giving an agent the semantics of the other agent. If an agent encounters another agent with semantics that it cannot understand, it should request help from a mutually understanding agent (or human expert) to acquire the semantics or even get the task done. Many who know much about the current structure of intelligent agents and Artificial Intelligence in general could argue that this is wishful thinking. However, the structure was developed and is presented in section 6.5.

8.5 Current GIS Issues and Improvements Through Agents: Geostatistical Pilot Study

GIS tools and services are usually large and cumbersome, often requiring human intervention for fine-tuning and/or to provide accurate answers (Isaaks and Srivastava 1989; Cressie, 1993; Bailey and Gatrell, 1995). Here, we have proposed that software agent(s) should be used to facilitate these issues. We have experimented using one of the GIS tools, the variogram, that is deemed to be very difficult to conceive without the help of a human expert (Isaaks and Srivastava 1989; Cressie, 1993). The main two problems regarding this research on GIS tools proved to be the size of the tools and data, the clustering (each tool being an expert of their function) and the constant requirement of the expert user - which takes us to the next problem, the time taken to actually perform a function. Here we identified features like intelligence, reactivity, mobility and social ability could help when dealing with this issue. However, on the agent side there were also issues in terms of developing an agent-based tool that was capable of achieving these capabilities. Thus, the experiment involved using a number of available development tools and programming languages. For this, Tropos with some improvements proved to be sufficient for the development of the GIS tool and similarly,

in implementation (programming), JACK proved efficient in most areas, but required much work in other areas - in the VAC case, mobility.

Geostatistics provides an easier mechanism to use, quantitatively analyse, characterise and predict the structure of the data under study, with its main tool being the variogram. However, the only way at the moment to produce a variogram is through the use of a GIS expert who has to physically visualise the data and decide upon the right mathematical functions to be incorporated in order to obtain the characteristics of the subject in question. For example, there may be a need to assemble data from across a network, a requirement to understand the nature of the data and its context, and to fit an appropriate model (e.g. Gaussian, spherical, quadratic; plus any antistrophic effects). Tool (different software systems) interoperability is also an important consideration in variogram construction and other aspects of geostatistics.

Another issue in this context is that, in geostatistics, human experts use quantitative techniques to analyse, characterise and predict the trend of the data under study. However, the need to analyse data qualitatively seems to be inevitable for prediction, such that geostatisticians have to always physically choose variables (such as the lag distance) to produce the quantitative analysis (Isaaks and Srivastava, 1989). As a result, to deal with the process of producing a variogram, Cressie (1993) identified the following tasks that an expert statistician is required to perform for the prediction:

1. design the sampling plan;
 2. graph and summarise the data;
 3. detect and allow for spatial non-stationariness;
 4. estimate spatial relationship, usually through structured analysis;
 5. estimate the *in situ* resources, usually through a validation technique like kriging;
- and

6. assess the resources and take decisions on the next action (e.g. mining); this should facilitate designing the execution plan and further observations of the resources.

The complexity of geographic data and processes raises other fundamental issues related to the incompatibility of representations, structures and semantics that need to be addressed to achieve geographic information Interoperability. The available literature indicates that several studies have been conducted on GIS and AI. Instead of teaching everybody to develop the GIS tool structure, it is surely better to have everybody do what they do best and simply have a Broker agent that could be fed with the semantics of GIS data (whether the data is an agent structure, object or relationship). The Broker agent could relate the current geostatistical functions which have been applied to that data (using the information that will be communicated by the agents). Such a structure would aid information access. The VAC is assumed to be:

- a continuous environment: infinite perception;
- dynamic: other external factors could possibly change the environment;
- inaccessible: cannot obtain complete, accurate information;
- non-deterministic: such that there is no single guaranteed effect, just like with human perception.

Thus, the variogram might realise the outcome of a run to be insufficient or maybe acquire extra data about the environment that would help in characterising the data. It should treat the situation according to time. It should also change the accuracy of its outcome by expecting a response from the next agent or environment. If this response has not been received, then it should enquire about the data. This response time to failure should be fairly short. So, after a decision has been made, the agent should expect a certain outcome, and thus listen to the effects or (when required) change the decision.

Furthermore, it has recently been argued that agent technology can substantially support the development of GIS in overcoming these kinds of limitations (Torrens, 2002; Reitsma, 2004; Albrecht, 2005; Batty, 2005a; Brown *et al.*, 2005). Multi-agent systems can be developed, where different agents can deal with different types of GIScience data in an effective way. Moreover, the autonomous abilities of multi-agent systems can reduce the workload currently required from GIScience experts. Here, the VAC is designed to deal with issues of handling large, distributed data by building better interoperability, extensibility and accessibility of GIS data analysis tools. As an experiment for agent functionality to provide a solution to these problems, a geostatistical tool (the variogram) was selected to run a pilot study as to the feasibility of using agents in GIS.

8.6 Critical Evaluation and Conclusion

8.6.1 Significance and Implication of the Findings

This section aims to establish alignment of the findings to the aims and objectives of the thesis. The research has been able to identify appropriate theoretical perspectives for analysing, developing and testing an algorithm model for an agent-based distributed GIS component. Developing an agent-based GIS as a service provider has been shown to be possible.

This continued with an examination of the existing agent environments and their architectures to determine which is best and fit for purpose to use as a variogram agent. This established the limitations and produced a suitable architecture (or the features to add to the architecture) to allow easy, flexible and efficient GIS agents. The appropriate software development methodology for analysis and design was established and argued. It was selected according to the currently available methodologies and the agent's functional requirement. The Tropos methodology was chosen. Areas which were deemed missing or weak (such as making it easy, like other methodologies, to understand the overall picture, provide an easy understanding of the analysis tools and

the lack of effective implementation diagrams) for this methodology were identified and solutions to these were outlined and experimented upon.

This exercise was repeated for the programming environment, where the implementation (programming) platform chosen was JACK. The process of identifying a suitable implementation environment was much more complicated than that of identifying the software development methodology. This was due to the fact that there is much more agent development at the moment than methodology development, and their functional capabilities are very different to one another (e.g. JACK would easily support intelligence but not mobility, while Egglet would support mobility but not intelligence). Examinations were performed to determine which one of the agent characteristics is harder to construct an algorithm for and implement, and which methodology supports it easily. In this case, intelligence seemed to be the key and it was concluded that the best implementation environment platform for this was JACK.

After identification of the appropriate software development methodology and implementation (programming) platform environment, it was necessary to identify the appropriate functional components for a Variogram agent. These were determined and incorporated into an agent system. These components were found to include cluster analysis, trend analysis and validation mechanisms. After the successful design using appropriate methodology and in accordance with the desired agent characteristics and features identified, and following the successful implementation of the VAC on the appropriate implementation environment, experimental tests were performed to test the validity of the newly developed VAC system. These tests were aimed at analysing its performance by validating its internal functionality, its perceived external output and support for agency due to its functional characteristics (those agent characteristics that give the VAC its personality). An internal validation was conducted to check for design and grammatical errors, whereby a few were found and rectified. Then an external validation was conducted to determine whether the agent had fully met its functional requirements and was able to provide an accurate analysis of the data and fit a

variogram. At this stage, tests were undertaken on nearest neighbour, cluster analysis, trend analysis, constructing variogram, fitting a variogram, validating a variogram (using Kriging and other methods) and other required procedures to achieve the desired results. The results were deemed sufficient and conclusions were drawn on the feasibility and practicality of using agent software for distributed component services in GIS.

8.6.2 Critical analysis

Most reference available defines conceptual design of agent system but implementation is often problematic to appropriate design and programming tools. Despite these issues, various authors have used agent technology in GIS system with success. The standing out literature of agent based GIS is Moulin *et al.* (2003), Benenson and Torrens (2004b), Batty (2005a), Torrens (2006 and 2007a) and Chaker *et al* (2009) where all have used agent technology to increase the system robustness and to allow simulation of environments. As an example SimWalk (2007) is a geosimulation-based experiment to enhance the security of public transport (train and bus) travellers. Whereas Nute *et al.* (2004) monitored the ecosystem of a forest using agent simulation. Similar experiment allowed Gosselin *et al.* (2005) to examine the expansion of the West Nile virus (WNV).

More process based agent were introduced by Purvis *et al.* (2003) to query and integrate distributed environmental information over a network. Similar research has been carried out using agent technology to control spatial data quality in geosimulation modelling by Li (2006) and more advance agent system demonstrated by Sahli and Moulin (2005) for Geo-simulation with agents capable of planning through anticipating a change of a scenario and by using agency characteristic of reactivity.

Furthermore, in chapter 3 agents were described by various role they play, object-entity, process, or service and categorised by the GIS data structure agents acting upon, tessellation or vector. Also within GIS community where agent use basic properties or

extra features as described on section 3.5 and demonstrated on table 6.1. On this context Heppenstall *et al.* (2005) described agent as entities exhibit basic features and working on tessellation and vector. The agents were able to communicate and optimise petrol pump functionality. Similarly Brown *et al.* (2005) provide the same capabilities but added processes into the agent functionality successfully. More service-oriented agents were described by Li, *et al.* (2008) albeit with basic agent features.

This thesis provides a concept of agency with extra features for assisting on reducing complexity when dealing with spatial data by utilising service based GIS agent, the VAC. These agents in VAC are developed as components capable of replicating themselves while adding or reducing knowledge of certain domain for the purpose of optimisation.

The challenge during the development of the VAC was that the software development tools inflexibility. The available Modelling languages and software development platforms for agents are non-standard and very difficult to align with the actual implementation process (Nikolai and Maddy 2009). These complications make it hard for novice software developers to be able to grasp the concept of agency and differentiate it with Object Oriented practices. Since GIS expert are often none-software development experts this hinders the potential of agents in solving GIS problems. This thesis try to improve the current tools to be more comprehensive in developing agent based GIS. The main limitation of this research is that the tool is tested specifically for mathematical based GIS solution and more qualitative test would be required in the future. Therefore, the VAC is only built as the basis of a theoretical framework that should enable these agent systems in GIS to act at as service rather than entities. However, successful test were done to show what agent systems are missing in support of GIS and these missing features were coded and tested. The results show success on the ability of easily generating agents that are more of service than entities or processes.

Among the important features is the ability of determining the resources available on an environment during agent creation (through reproduction, cloning or compilation). As an

agent is created it will request for dedicated processor space (semaphore). Using this knowledge it will assess its processing capabilities and make sure any task it agrees to undertake that is larger than its capability it will address it by using file segmenting techniques explained in section 7.4. Furthermore, the VAC periodically (every 12 runs) re-assesses the resources and makes sure to re-tune in case of memory degradation.

The main challenges to the VAC and its development process was the testing of the process since it is developed in resource-controlled environment (set from its initiation). Another challenge was to determine when the human user can/should stop interacting with the system.

The main advantage is that it provides knowledge sharing environment, speeds up the GIS process and eliminates large dataset issues (crash of the system, etc.), even if the human expert is involved all the time these advantages still makes the VAC system very desirable.

8.6.3 Conclusion

This thesis has presented a novel multi-agent system to support the field of GIS in general and the analysis of geostatistics in particular. This system is the first attempt to develop a multi-agent system for a service based distributed component GIS. As explained on section 3.2 the service is based on agent characteristics such as intelligence, autonomicity, mobility and reactivity. The discussion on section 3.5 and the analysis given on section 8.6.1 show that GIS community have realised the potential of agent technology and have been studying and implementing them into the GIS domain. However, as shown on these section most of the research is on the potential of agent as entities, while only few have been using agents as process. This research has shown the potential of agent embedded as GIS service. An important aspect of the system is the ability to log the geostatistical tools used for the decisions it made during the analysis and so reduces the GIS expert's workload. This is very important, since it allows the

system to report such information to the human expert if the results obtained are not satisfactory. By having input from the human expert, the system is able to reverse the analysis order and identify the stage at which the error occurred.

Other methods that are used to rectify a decision are achieved by the VAC checking on itself, such that it works using its previous logs and, if the results are expected to trigger a certain action (in this case, it could be producing a certain graph or table, or even just a sentence of decision) and this action does not take place, then reverse working from an action point of view can be undertaken so that the point that caused the difference can be noted into the log. It is important that, at this point in the process, the agent can receive feedback on the actions, so that it can decide what was wrong with the decision (if the decision was wrong) or whether it was a different phenomenon with similar data characteristics to other previous processes. This will help to fine tune the agent's knowledge.

Accuracy of the system has been validated by using a well-known set of GIScience data, then comparing the results of the VAC with the published results from the same data using a similar analysis (Isaaks and Srivastava, 1989). The results were encouraging, since the VAC produced the same output as the original analysis from the human experts. The tests were performed using real life data and the technology and its architecture proved to be useful and helpful in achieving the intended requirements and results.

The system demonstrates the possibility of having agent provide service to GIS. The main finding was the need of new function to support agent characteristics to be incorporated into the current agent platform to achieve desired results. First cloning needed to be extended, since it only replicates the agent and does not address the issue of unique analysis required for GIS data. Hence, reproduction characteristic was implemented and tested (section 7.4.2) as an extension of cloning in JACK platform. For this function to be effective its processes and calculations had to be dynamically bound

to a running agent. Therefore a function to dynamically bind the information (process) was also designed, implemented and tested. Additionally, the communication for distributed component GIS had to be on a specialisation basis (due to uniqueness of individual GIS data) brokerage mechanism for data and process was established. After these features implementation the actual analysis of the system was done. The agent was able to function as a component and when compared to existing service based GIS system it had much better performance, the performance was also compared to human expert with superior effect (shown on section 7.4).

This is a first attempt to develop a deployable system of loosely coupled GIS components based on agent technology. These components should be able to make GIS perform large process in shorter time with high data integrity and less resource constraint. A potential of these components will be demonstrated on the following sections. Nevertheless, more work is needed, in particular with respect to knowledge definition. Knowledge definition, in this case, refers to an agent's ability to understand the dynamically bound plans (formulae and processes within a plan) and create a mechanism for re-utilising these plans. This mechanism aims to ultimately make the agent more autonomous, intelligent and responsive. However, this function was successfully implemented and tested. Discussions regarding the performance and future of this agent and software agents for GIS in general were presented.

8.7 Future Work

There are many functionalities, improvements and tests that could be introduced into agent-based GIS systems. Focusing initially on the VAC, the first step is to provide the ability to construct maps for all analysis outcomes from provided data. For example, when the dynamic binding feature of the agent determines that the dataset contains clusters, it would be useful to be able to present these clusters on a map for easier and quicker analysis.

Regarding the dynamic binding feature, the system should be improved to allow agent to agent binding, whereby once an agent has analysed the dataset it could then ask other agents for their opinion (and any rectification) before it asks the human user. The reproduction function (described in Chapter Six) should be fully implemented and tested, to assist in this function. Furthermore, with regard to dynamic binding, the agent should allow for more analysis, rather than simply asking the user to input the new functions. This would ensure that during the learning period, the expert would suggest and state whether the agent's result is right or wrong. The suggestions would be of a similar pattern to those used for NED-2 by Nute *et al.* (2004), but would be in more detail and made easier to enable this information to be added to the knowledge log of the individual agent. Here, the choices will be:

- Right - completely right.
- Wrong - completely wrong, try again.
- Almost Right - it is ok, but could improved.
- Almost Wrong – it is ok, but should be improved.
- About OK – it's in-between.

Having received the data in a 'churned up' manner from an external source like remote sensing, an agent could communicate with external vendors to negotiate for an appropriate image for the study at hand. The agent should use a tool (such as a scatter plot) to clear the data and save the effects (statistical formulae and data properties) that have been noted during the cleansing function into a data file. However, the knowledge acquired during this run would need to be saved in a symbolic format, so as to maintain the GIS data heterogeneity and tool interoperability. The agent could use these symbols to interpret what had happened to the data when applied to this statistical analysis, and could also identify any similarities of the final statistics to the data pattern (Issak and Srivastava, 1989).

Table 8.1: Knowledge Structure or Agent System

Formula	Outcome	Symbol
1	Right	1
2	Almost Right	3
3	Wrong	2
4	Almost Wrong	4
5	About OK	5

The next time that this type of data or process type is recognised, the agent can increase the weight of the symbol and a confidence level will be noted. In addition, each time such a pattern is recognised then the process can commence from the recorded point where it was correctly started the previous time(s). With a certain amount of confidence, it will be able to do fewer runs and, eventually, it should be able to understand the data pattern and use only one formula in order to produce the scatter plot for the data. This will improve the intelligence, by creating better look-up table mechanisms. Furthermore, this could be incorporated with other AI techniques, particularly for pattern recognition and neural networks.

The VAC should also be implemented into geosimulation engines to study its support of the increasing functionality of other successful agent-based systems in GIS. To effectively achieve this goal, more variogram construction functions should be defined and implemented into the VAC (such as introducing more variogram models) and other GIS components should be designed, implemented and tested to allow wider collaboration between GIS tools. A model (similar to that shown in Figure 8.1) should be established. This model would allow for GIS improvements in data sharing and fetching.

8.7.1 Use of the VAC With Other Agent Systems in GIS

The VAC should be implemented to support and increase the functionality of other successful agent-based systems used in GIS; in particular in geosimulation where agents are currently being utilised to help improve data analysis. The VAC should be used in the geosimulation environment as a service provider. This should be a mechanism to take over from the human user who would undertake an analysis after the data has been made rich by the agent simulation process in the geosimulation. Moreover, components of different functions should be implemented and utilised for this kind of support.

Introducing the VAC into geosimulations is the next step of the thesis. The mechanism that shows the position of the VAC and other agent components that provide service to GIS, the Broker Agent Component (BAC) and the Map Agent Component (MAC), is given in Figure 8.2.

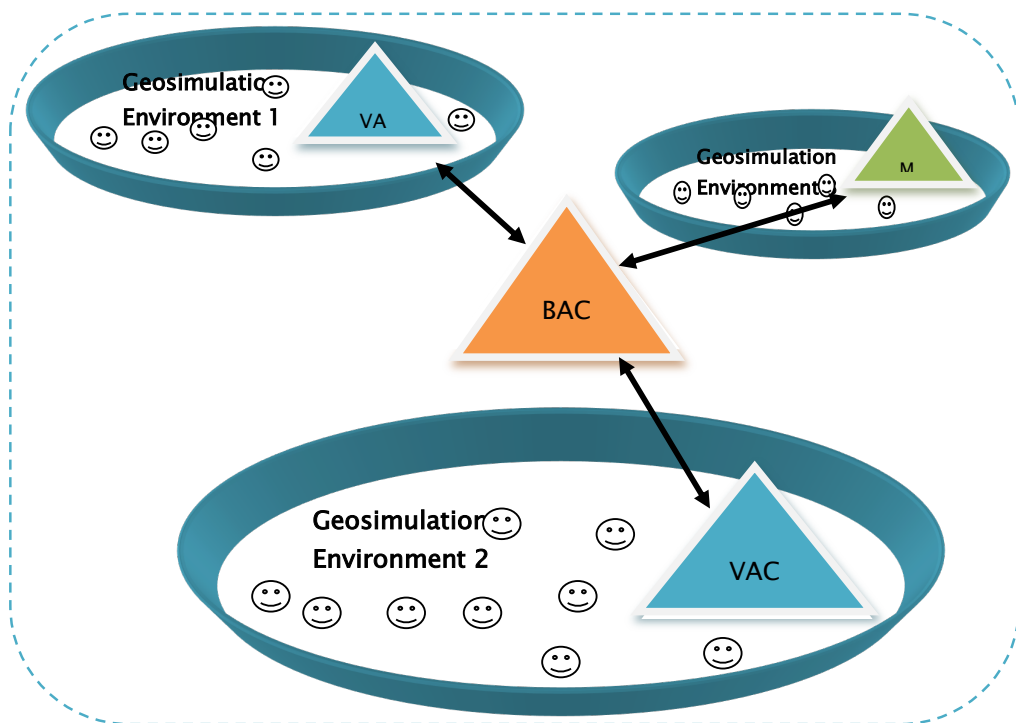


Figure 8.1. Agent as Service Provider to Geosimulations

In geosimulations, agents are used as interacting entities that model real life objects and provide scenarios of data function in their perceived environment. More functions exist for geosimulations which do not include agents as service providers. For agents to function as service providers, a very defined and specialised architecture is required, like that given by the VAC. In the geosimulation environment, the VAC could be used to provide services to other GIS tools, like defining dependency within entities of a particular environment and building a relationship of entities. Furthermore, the architecture provided by the VAC could be extended to other services (like the example given in Figure 8.2) and more functions introduced where maps are provided. These functional agents will act like other internal agents in the geosimulation, but with the specialised task of providing services to the environment. This architecture can be further extended by providing a Broker agent component to help provide a service from one geosimulation environment to another which does not support that service but has a need for it. As per agent simulation theories (Batty, 2005b) this mechanism should also provide a bigger and more robust geosimulation system in a distributed environment using the Broker agent components as the communicating agent (see Figure 8.1 for details).

8.7.2. Communication of Distributed Component GIS

Using the agent simulation will provide an access point where agents of other GIS components can be integrated and thus provide a communication channel between the multi-components of GIS. This will draw the structure to introduce GIS into a real ubiquitous computing paradigm, where GIS components can loosely couple and form an better services rather than rely on multiple remote components that have to be called on to perform a function when they are needed. This is shown in Figure 8.2 with an example of communication of multi-component GIS using information resulting from the VAC for a remote sensing image that needs to be interpolated.

The structure is such that each GIS component is built as an autonomous agent that is designated to its goal - and only its personal goal. Here, the negotiator agent compares the price and resources of each remote sensing vendor and recognises (according to the need) which image is the best fit for the purpose and, in turn, acquires it. The negotiator would keep track of this information, so the next time the same or similar information is requested it would contact the appropriate vendor without losing time for comparing. This technique provides for easy extensibility and learning mechanisms.

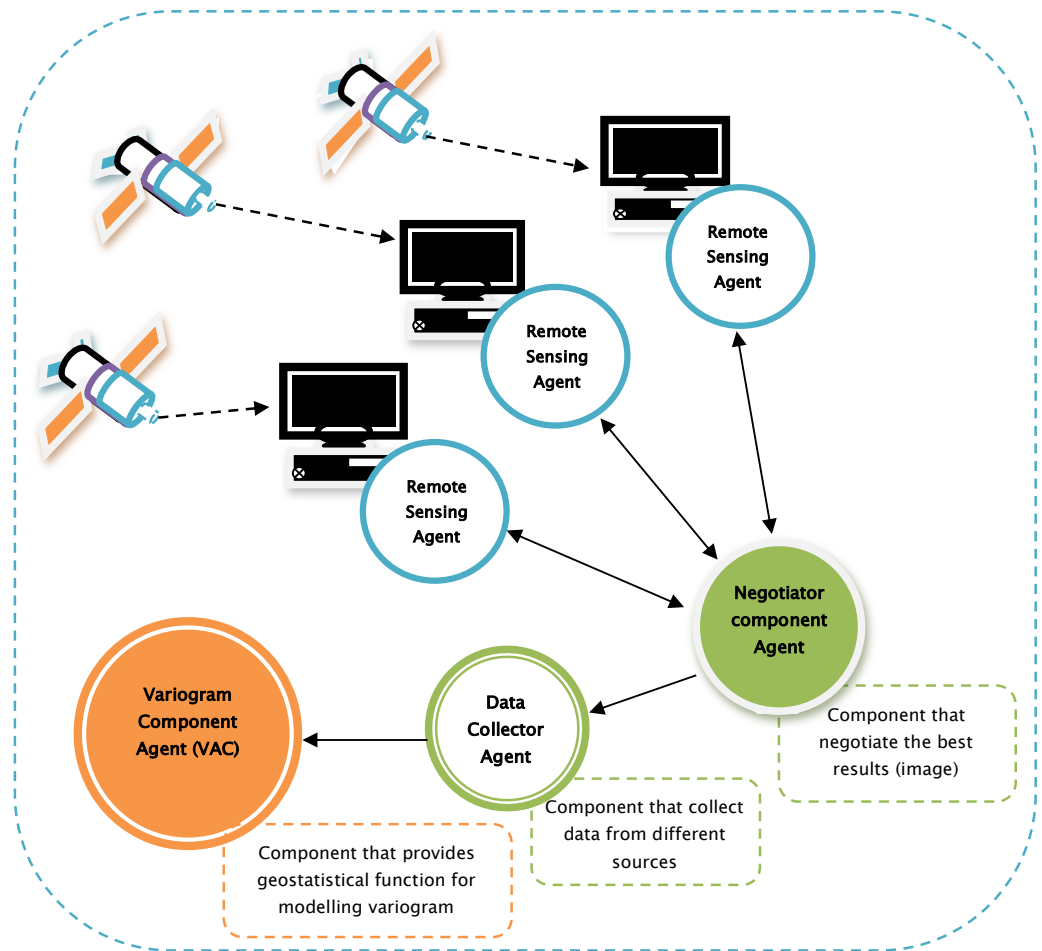


Figure 8.2: Agent Communication and Information Sharing

8.7.3 Expanding Distributed Component GIS

On analysis, the requirement is that the agent should be extended to be able to fetch data through a network and implement the proactive characteristic of the software agent. There should not be a predetermined data location (i.e. the VAC can only fetch from these pre-determined locations), but the agent should take the initiative to fetch data from all sources that it has access to. This fetching data function should allow an agent to determine the best data for the analysis (for example, if a dataset is required to analyse London's transportation patterns, the agent should be able to go to different sources [data owners]). Furthermore, if the data that was acquired by the agent proves to have a problem in it, then the agent should be able to look for other (cheaper) mechanisms to rectify that problem. In some circumstances, the agent might determine that it is better to buy only the section of the data that is causing the problem and not the whole dataset. This should improve the analysis, reduce the time taken to handle the data and, most importantly, reduce the cost of analysis. This would improve the function of high performance GIServices (Friis-Christensen *et al.*, 2007) and promote GIS as being an easy to use service and ensure its widespread use via technologies such as the Short Message Service (SMS) of mobile telephones.

Furthermore, test should be conducted to determine the limitation of the VAC as a tool. In this research agency functionality and the development of the agent based GIS was the focus and extensive tests were done but the deployment of the actual tool into a larger environment (in terms of network and data) to test its versatility.

REFERENCES AND BIBLIOGRAPHY

- A2D (2004). "Advanced Well Log Data Products". Available on-line at <http://www.a2d.com/default.asp>, accessed 12-12-04.
- Abdallah, J., Adamick, J. and Helland, A. (2004). "TGS-NOPEC Expands Gulf of Mexico Multi-Client 3D Project". Available on-line at, http://www.tgsnopec.no/whatsnew/GOM_MC3D_Project.htm, accessed 12-12-04.
- Abelson, H. and Sussman, G. (1984). *The Structure and Interpretation of Computer Programs*, Massachusetts, MIT Press.
- Ahlqvist, O., Bibby, P., Duckham, M., Fisher, P., Harvey, F. and Schuurman, N. (2005) Not Just Objects: Reconstructing Objects, In Fischer, P. and Unwin, D. *Representing GIS*, London, John Wiley & Sons, pp.17-26
- Akgun A, Dag S, Bulut F (2008) Landslide susceptibility mapping for a landslide-prone area (Findikli, NE of Turkey) by likelihood-frequency ratio and weighted linear combination models. *Environ Geol* **54**(6)1127–1143
- Al-Zakwani, A. (2006) Mobile Devices Evolution and Revolution: A Cause for Security Concern. *International Conference on Internet Computing* **2006**: 369-376
- Al-Zakwani, A. (2007a) Establishing a platform for networked games on mobile devices using SMS/GPRS, *Proceedings of Advances in Computing and Technology, (AC&T) The School of Computing and Technology 2nd Annual Conference*, University of East London, pp.128-134
- Al-Zakwani, A., Brimicombe, A. and Mouratidis, H. (2007b) An Agent-Based System to Support Geo-Information Analysis. *IAT* **2007**: 269-272
- Al-Zakwani, A., Mouratidis, H. and Brimicombe, A. (2007a) "An agent-based system to support geo-information analysis" *Proceedings Intelligent Agent Technology (IAT 2007)*, Silicon Valley, CA: 269-272

- Al-Zakwani, A., Mouratidis, H. and Brimicombe, A. (2007b) "A dynamic binding technology for agent-based geo-information systems" *Proceedings of Intelligent Systems & Agents* (IADIS 2007), Lisbon, Portugal: 117-123 [note: Outstanding Paper Award]
- Albrecht, J. (1996). *Universal GIS-operations: A Task-Oriented Systematization of Data Structure-Independent GIS Functionality Leading towards a Geographic Modelling Language*. Vechta, Insitut fur Strukturforschung and Planung in Agrarischen Invtensivegebieten.
- Albrecht, J. (2005). A new age for geosimulation, *Transactions in GIS*, **9**(4), 451-454.
- Albrecht, J. (2007). *Key Concepts and Techniques in GIS*, London, Sage Publications.
- Albrecht, J. (2008). *Dynamic Modelling*. In: J. Wilson & S. Fotheringham (eds.), *The Handbook of Geographic Information*, Malden MA, Blackwell, pp.436-446.
- Alexander, C. (1977). *A Pattern Language*, Oxford, Oxford University Press.
- Ali, W. and Moulin, B. (2005). 2D-3D MultiAgent GeoSimulation with Knowledge-Based Agents of Customers' Shopping Behavior in a Shopping Mall, *COSIT* (2005), 445-458.
- Allison, B. (1997). *The Student's Guide to Preparing Dissertations and Theses*, London, Kogan Page.
- Alston, R. and Donelan, D. (1992). A GIS is a Terrible Thing to Waste! A Comprehensive Cost-Benefit Analysis of Geographic Information Systems, *Proceedings of the 20th Annual AURISA Conference*, Gold Coast, **1**, 390-394.
- AM (2004). "What is geoscience?", Australian Museum. Available on-line at <http://www.amonline.net.au/geoscience/about/index.htm>, accessed 02-07-2005.
- Amandi, A., Campo, M. and Zunino, A. (2005). JavaLog: a framework-based integration of Java and Prolog for agent-oriented programming, *Computer Languages, Systems and Structures*, **31**, 17-33.

- Amirian, P. and Mansurian, A. (2006). Potential of Using Web Services in Distributed GIS Applications, *GIS Development, Middle East: Natural Resource Management/Land Administration/Web GIS*, **2**(6), 1.
- AMS (2006). "Glossary of Meteorology", American meteorological Society. Available on-line at <http://amsglossary.allenpress.com/glossary/browse?s=n&p=28>, accessed 23-11-2006
- AMS (2006). [*American Meteorological Society*](#), *Glossary of Meteorology*, Allen Press. Available on-line at <http://amsglossary.allenpress.com/glossary/browse?s=n&p=28> accessed on 18/02/2006
- An, L., Linderman, M., Qi, J., Shortridge, A. and Liu, J. (2005). Exploring complexity in a human environment system: An agent-based spatial model for multidisciplinary and multi-scale integration, *Annals of the Association of American Geographers* **95**, 54–79.
- Anderson, N. (2006). "nobody even knows what it means", Tim Berners-Lee on Web 2. Available on-line at www.arstechnica.com, accessed 2006-09-05.
- Anon (2004). "GIS Data Modeling and ANALYSIS". Available on-line at http://www.tgsnopec.no/whatsnew/GOM_MC3D_Project.htm, accessed 12-12-04
- Anselin, L. (1988). *Spatial Econometrics: Methods and Models*, Dordrecht, Kluwer Academic Publishers.
- Anselin, L. (1992). SpaceStat, a Software Program for Analysis of Spatial Data, *National Center for Geographic Information and Analysis (NCGIA)*, University of California, Santa Barbara, California.
- Anselin, L. (1994). Local Indicators of Spatial Association—LISA, *Geographical Analysis*, **27**, 93–115.
- Anselin, L. (1995). Local indicators of spatial association — LISA, *Geographical Analysis*, **27**, 93–115.

- Anselin, L. (2005). Exploring Spatial Data with GeoDATM: A Workbook, *Spatial Analysis Laboratory*, pp. 138.
- Anselin, L., Syabri, I. and Kho, Y. (2006). GeoDa: An introduction to spatial data analysis, *Geographical Analysis* **38**, 5-22.
- APA (1996). Intelligence: Knowns and Unknowns American Psychologist, February addition 1996, *Official Journal of the APA*.
- Armaly, B., Durst, F., Pereira, J. and Schonung, B. (1983). Experimental and theoretical investigation of backward-facing step flow, *Journal of Fluid Mechanics*, **127**, 473–496.
- Atkinson, P. and Tate, N. (2000). Spatial scale problems and geostatistical solutions, *Professional Geographer*, **52**(4), 607-623.
- Bailey, T. and Gatrell, A. (1995). *Interactive spatial data analysis*, Essex, Longman Scientific and Technical.
- Bailey, T. and Gatrell, A. (1998). *Interactive spatial data analysis (2nd edition)*, Essex, Addison Wesley Longman.
- Balce, A. (1987). Determination of optimum sampling interval in grid digital elevation models (DEM) data acquisition, *Photogrammetric Engineering and Remote Sensing*, **53**, 323-330.
- Ball, D. and Babbage, R. (1989). *Geographic Information Systems, Defence application*, Australia, Pergamon Press.
- Barnes, R. (2002) *Variogram Tutorial*, Golden Software, Inc. USA
- Barnwal, R. (2007). "Web 2.0 is all about understanding the economic value of social interaction". Available on-line at www.alooTechie.com, accessed 02-23-02-2007.
- Barrass, R. (1991). *Scientists Must Write*, London, Chapman and Hall.

- Barry, R. (1996). A Diagnostic to Assess the Fit of a Variogram Model to Spatial Data, *Journal of Statistical Software*, **1**(1), 1-11.
- Barry, T. (1996). Recommendations on the testing and use of pseudo-random number generators used in Monte Carlo analyses for risk assessment, *Risk Analysis*, **16**(1), 93-105.
- Batty M. and Xie Y. (2003). "Integrated urban evolutionary modelling", CASA working paper 68. Available on-line at http://www.casa.ucl.ac.uk/working_papers, accessed on 11-11-2006.
- Batty, M and Torrens P. (2001). "Modeling Complexity: The limits to Prediction", CASA working paper 36. Available on-line at www.casa.ucl.ac.uk, accessed on 11-02-2006.
- Batty, M. (2000a). The new urban geography of the third dimension, Environment and Planning B, *Planning and Design* **27**, 483 – 484.
- Batty, M. (2000b). *GeoComputation using cellular automata*. In: Openshaw S and Abrahart R, (eds.), *GeoComputation*, London, Taylor and Francis, pp. 95 – 126.
- Batty, M. (2001a). Polynucleated urban Landscapes, *Urban Studies*, **38**(4), 635-655
- Batty, M. (2001b). Models in planning: technological imperatives and changing roles, *International Journal of Applied Earth Observation & Geoinformation* **3**(3), 252–266
- Batty, M. (2005a). Agents, cells and cities: Representational models for simulating multiscale urban dynamics, *Environment and planning A*, **37**, 1363-1394
- Batty, M. (2005b). *Cities and Complexity Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals*, Massachusetts, MIT Press.
- Batty, M. (2005c) Network Geography: Relations, Interactions, Scaling and Spatial Processes in GIS, In Fischer, P. and Unwin, D. Re-presenting GIS, John Wiley & Sons: England, pp. 149-170

- Batty, M. and Longley, P. (1994). *Fractal cities, a Geometry of Form and Function*, London, Academic Press.
- Batty, M. and Torrens, P. (2001b). Modeling complexity: the limits to prediction, *CyberGeo*, 201.
- Batty, M. and Torrens, P. (2005). Modelling and prediction in a complex world. *Futures*, **37**, 745–766
- Batty, M., Chapman, D., Evans S., Haklay M., Kueppers S., Shiode N., Smith A. and Torrens P. (2001). *Visualizing the city: communicating urban design to planners and decision-makers, Planning Support Systems: Integrating Geographic Information Systems, Models, and Visualization Tools*. In: R. Brail and R. Klosterman (eds.), *Redlands, CA and New Brunswick*, New Jersey, ESRI Press and Center for Urban Policy Research Press, pp. 405-443.
- Batty, M., Desyllas, J., and Duxbury, E. (2003). The discrete dynamics of small-scale spatial events: agent-based models of mobility in carnivals and street parades, *International Journal of Geographic Information Science*, **17**, 673-697.
- Batty, M., Galton, A. and Llobera, M. (2005) Not Just Space: An Introduction, In Fischer, P. and Unwin, D. *Re-presenting GIS*, London, John Wiley and Sons, pp. 127-134
- Batty, M., Helen, C. and Eichen, M. (1997). Special issue: urban systems as cellular automata, *Environment and Planning B*, **24**(2).
- Batty, M., Xie, Y. and Sun, (1999). "The dynamics of urban sprawl", CASA working paper 15. Available on-line at http://www.casa.ucl.ac.uk/working_papers, accessed on 11-02-2006.
- Bédard, Y., Gosselin, P., Rivest, S., Proulx, M., Nadeau, M., Lebel, G. and Gagnon, M. (2003). Integrating GIS components with knowledge discovery technology for environmental health decision support. *International Journal of Medical Informatics*, **70**(1), 79-94.
- Bédard, Y., P. Gosselin, P., Rivest, S., Proulx, M., Nadeau, M., Lebel G. and Gagnon, M. (2003). Integrating GIS Components with Knowledge Discovery Technology for

Environmental Health Decision Support, *International Journal of Medical Informatics*, **70**(1), 79-94

Bell, J. (2002). *Doing your research project (3rd edition)*, Buckingham, Open University Press.

Benenson, I. and Torrens, P. (2004b). *Geosimulation: Automata-based Modeling of Urban Phenomena*, Chichester, John Wiley.

Benenson, I. and Torrens, P. (2004a). Geosimulation: object-based modelling urban phenomena, *Computers, Environment and Urban Systems*, **28**(1-2), 1-8.

Bennett, D., Wade, G. and Armstrong M. (2002). Exploring the solution space of semi-structured geographical problems using genetic algorithms, *Transactions in GIS*, **3**(1), 51-71.

Bennett, D., Wade, G. and Sengupta, R. (1999) Geographical Modeling in Heterogeneous Computing Environment, In M. Goodchild, M. Egenhofer, R., Fegeas, and C. Kottman, Eds.) *Interoperating Geographic Information Systems*, Dordrecht, Kluwer Academic Publishers, pp. 149-164

Berners-Lee, T. (2006). "developerWorks Interviews: Tim Berners-Lee". Available on-line at <http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html>, accessed on 12/12/2006.

Berry, B., Griffith, D. and Tiefelsdorf, M. (2008). From spatial analysis to geospatial science. *Geographical Analysis*, **40**, 229-238.

Best, D. (2006). Web 2.0 Next Big Thing or Next Big Internet Bubble? Lecture: Web Information Systems, *Technische Universiteit Eindhoven*.

Bevan, K. and Binley, A. (1991). The future of distributed models: model calibration and uncertainty prediction, *Hydrological Processes*, **6**, 279–298.

Bhatt, G., Kumar, M. and Duffy, C. (2008). Bridging the Gap between Geohydrologic Data and Distributed Hydrologic Modeling, *iEMSs 2008: International Congress on Environmental Modelling and Software Integrating Sciences and Information*

Technology for Environmental Assessment and Decision Making 4th Biennial Meeting of iEMSs.

- Bian, L. and Walsh, S. (1993). Scale dependencies of vegetation and topography in a mountainous environment of Montana, *Professional Geographer*, **45**, 1-1.
- Bigus, J. and Bigus, J. (2003). *Constructing Intelligent Agents Using Java (2nd edition)*, New York, John Wiley & Sons, Inc.
- Bingham, J, Cuthbert, L., Yang, X., Ning, L. and Ryan, D. (2004). Using intelligent agents for managing resources in military communications, *Computer networks*, **46**, 709-721.
- Birkin, M., Wu, B. and Rees, P. (2009) Moses: Dynamic Spatial Microsimulation with demographic interactions, In: Zaidi A; Harding A; Williamson P (Ed) *New frontiers in microsimulation modelling*, Ashgate, pp.53-78.
- Blaxter, L., Hughes, C. and Tight, M. (2002). *How to research (2nd ed.)*, Buckingham, Open University Press.
- Blaxter, L., Hughes, C., Tight, M. (2002). *How to Research (2nd edition)*, Berkshire, Open University press.
- Bobrow, D. (1964). "Natural language input for a computer problem-solving system". M.I.T, Ph.D. Theses.
- Bock, C. and Odell, J. (1998). A More Complete Model of Relations and Their Implementation: Roles, *Journal of Object-Oriented Programming*, **11**, 51-54.
- Bohling, G. (2005). "Introduction to Geostatistics and Variogram analysis". C&PE 940. Available on-line at <http://people.ku.edu/~gbohling/cpe940>, accessed 17-12-2005.
- BotSpot (2007). <http://www.botspot.com>, accessed on 23-01-2007.
- Bouden, M., Moulin, B., Gosselin, P., Back, C., Doyon, B., Gingras, D. and Lebel, G. (2005). The Geosimulation of West Nile Virus Infection on the Basis Of Climate: A

Tool for Risk Management in Public Health. *Proceedings for C-CIARN Conference*, Montreal, Canada.

Box, G. and Cox, D.(1964). An Analysis of Transformations, *Journal of the Royal Statistical Society*, **26**, 211-246.

Box, G., and Jenkins, G. (1976). *Time Series Analysis: Forecasting and Control*, Oakland, Holden-Day.

Box, G., Hunter, W. and Hunter, J. (1978). *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*, New York, John Wiley and Sons.

Brandmeyer, J. and Karimi, H. (2000). Coupling Methodologies for Environmental Models. *Environmental Modeling & Software*, **15**(5), 479-488.

Braswell, R, Lawyer, G, Cluff, B, Upchurch, W, Lefever, J, (2000). Horizontal drilling in oil shales. *Proceedings for PTTC's Eastern Gulf Region*, Jackson, Mississippi.

Bratman, M. E. (1999) *Intention, Plans, and Practical Reason*. CSLI Publications, Stanford

Bray, M. (1997). "Object-Oriented Programming Languages: Software technology road map", Carnegie Mellon University. Available on-line at http://www.sei.cmu.edu/str/descriptions/oopl_body.html, accessed 23-04-2005.

Brazier, F., Dunin-Keplicz, B., Treur, J., and Verbrugge, L. (1999). Modelling the internal behaviour of BDI-agents. In: J.-J. Ch. Meyer and P.Y. Schobbès (eds.), *Formal Models of Agents (Selected papers from final Model Age Workshop)*, *Lecture Notes in AI*, Springer Verlag, **760**, 36-56.

Brazier, F., Jonker, C. and Treur, J. (2000) Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal*, **14**(2000), 491-538.

Brazier, M., Jonker, M. and Treur, J. (2002). Principles of component-based design of intelligent agents, *Data & Knowledge Engineering*, **41**, 1–27.

Brebbia, C., Walker, S. (1979). *Dynamic analysis of offshore structures*, London, Newness-Butterworths.

- Bresciani, M. J., Zelna, C. L., & Anderson, J. A. (2004). Assessing student learning and development. A Handbook for Practitioners. United States: NASPA.
- Bresciani, P. and Sannicolò, F. (2002). Applying Tropos Requirements Analysis for defining a Tropos Object Oriented. *Agent-Oriented Information System, AOIS-2002: Fourth International Bi-Conference Workshop*, Toronto, Canada.
- Bresciani, P., Giorgini, P., Henderson-Sellers, B. (2003). Evaluating the Potential for Integrating the OPEN and Tropos Metamodels. *International Conference on Software Engineering Research and Practice (SERP'03)*.
- Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A. (2004). TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers.
- Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A. (2000). Modeling early requirements in Tropos: a transformation based approach, *Second International Workshop on Agent-Oriented Software Engineering (AOSE-2001)*. Montreal, Canada.
- Brimicombe, A. (1998). A fuzzy set approach to using linguistic hedges in geographical information systems, *CyberGEO - European Journal of Geography*.
- Brimicombe, A. (2000). Encoding expert opinion in Geo-Information System: a fuzzy set solution, *Transaction in Internal Land Management*, **1**, 105 – 121.
- Brimicombe, A. (2003). *GIS, Environmental Modelling and Engineering*, London, Taylor and Francis.
- Brimicombe, A. (2008). Location-Based Services and Geographic Information Systems. In Wilson, J.P. and Fotheringham, S. (eds.) *The Handbook of Geographic Information Science*, Malden MA, Blackwell: 581-595
- Brimicombe, A. and Li, C. (2009) *Location-Based Services and Geo-Information Engineering*. Wiley, Chichester.

- Brimicombe, A. and Li, Y. (2006). Mobile Space-Time Envelopes for Location-Based Services, *Transactions in GIS*, **10**(1), 5-23.
- Brimicombe, A. J. and Li, C. (2009) *Location-Based Services and Geo-Information Engineering*. Boca Raton, CRC Press.
- Brimicombe, A., Li, Y., Al-Zakwani, A. and Li, C. (2009) Agent-Based Distributed Component Services in Spatial Modeling, In *Computational Science and its Applications - ICCSA 2009* (eds. Gervasi et al.) Berlin, Springer, pp.300-312
- Brown, D., Riolo, R., Robinson, D., North, M. and Rand, W. (2005). Spatial process and data models: Toward integration of agent-based models and GIS, *Journal of Geographical System*, **7**, 25-47.
- Brown, G., Page, S., Riolo, R. and Rand, W. (2004). Agent based and analytical modeling to evaluate the effectiveness of greenbelts, *Environmental Modelling and Software*, **19**, 1097-1109.
- Brown, S., McDowell, L., Race, P. (1995). *500 tips for research students*, London, Kogan Page Ltd.
- Buehler, K. and McKee, L. (1998), The OpenGIS Guide - Introduction to Interoperable Geoprocessing and the OpenGIS Specification. 3rd Edition; *OpenGIS Consortium*, Wayland, Massachusett
- Bui, E. (1999). A soil information strategy for the Murray-Darling Basin, *Report to Murray Darling Basin Commission*, Project D5038.
- Bukovics, B. (2006) *.NET 2.0 Interoperability Recipes: A Problem-Solution Approach (Expert's Voice in .NET)*, New York, Springer-Verlag.
- Burden, P. and Parker, C. (1994). The Ministry of Water Resources Geographical Information System, 1: *Main Report, Unpublished internal report*.
- Burrough, P. (2000). Whither GIS (as system and science)?, *Computers, Environment and Urban systems*, **24**, 1 – 3.

- Burrough, P. and McDonnell, R. (1998). *Principles of Geographical information systems*, Oxford, Oxford University Press.
- Bybee, K. (2004). Underbalanced Drilling With Casing in South Texas. *Shell Intl. and E&P Inc: prepared for the 2003 SPE Annual Technical Conference and Exhibition*, Denver.
- Caers, J. (2001). Geostatistical reservoir modelling using statistical pattern recognition, *Journal of Petroleum Science and Engineering*, **29**, 117-188.
- Callan, R. (2003). *Artificial Intelligence*, London, Palgrave Macmillan.
- CdMMF (2005). "Centre de Morphologie Mathématique in Fontainebleau, France". Available on-line at http://cg.enscm.fr/Presentation/Matheron/Matheron_en.shtml, accessed 23-11-2005.
- Chaker, W., Proulx, M., Moulin, B. and Bédard, Y. (2009) Modélisation, simulation et analyse d'environnements urbains peuplés Approche multi-agent pour l'étude des déplacements multimodaux, *Revue Internationale de Géomatique*, **19**(4), 413-441.
- Chambers, J., Cleveland, W., Kleiner, B. and Tukey, P. (1983). *Graphical Methods for Data Analysis*, Monterey, Wadsworth.
- Chambers, J.; Cleveland, W.; Kleiner, B. & Tukey, P. (1983), 'Graphical Methods for Data Analysis', *The Wadsworth Statistics/Probability Series*. Boston, MA: Duxury
- Chan, H. and Grosz, B. (2008). A rule-based framework for developing rule-based applications with major emphasis on maximum separation of business logic and data, conflict handling, and interoperability of rules. <http://www.alphaworks.ibm.com/tech/commonrules> accessed Nov 22, 2008,
- Charatan, Q. and Kans, A. (2002). *Java in two semesters*, London, Prentice Hall.
- Chatterjee, S. and Webber, J. (2004) *Developing Enterprise Web Services: An Architect's Guide*, New Jersey, Prentice-Hall.

- Chu, A., Kwok, R., Yu, K. and Short, (2005). Communication study of pollution dispersion in urban areas using Computational Fluid Dynamics (CFD) and Geographic Information System (GIS), *Environmental Modelling & Software*, **20**, 273–277.
- Chua, L., Merting, F., Holz, K. (2002). Numerical Simulation and Web-based GIS System for Flooding Processes. *ICHE conferences*, Warsaw.
- Clementini, E., Di Felice, P. & van Oosterom, P. (1993). A small set of formal topological relationships suitable for end-user interaction. In: D. Abel & B. Ooi (eds), *Advances In Spatial Databases*, Springer-Verlag, pp. 276–295.
- Cliff, A.D., and J.K. Ord. (1981) *Spatial processes: models and applications*, Taylor & Francis.
- Coffey, J., Can˜as, A., Hill, G., Carff, R., Reichherzer, T. and Suri, N. (2003). Knowledge modeling and the creation of EI-Tech: a performance support and training system for electronic technicians, *Expert Systems with Applications*, **25**, 483–492.
- Cohen, L. (1976). *Educational research in classroom and schools: A manual of materials and methods*, London, Harper and Row.
- Connolly, T. and Begg, C. (2004). *Database Systems A Practical Approach to Design, Implementation, and Management (4th edition)*, New Jersey, Addison-Wesley.
- Conrad, L. (1996). Varying approaches to academic research development. Different Approaches: Theory and Practice in Higher Education. *Proceedings HERDSA Conference*, Perth, Western Australia.
- Cowen, J. (1998). *On becoming an Innovative University teacher; Reflection in action*, SRHE, Open University Press.
- Crème, P., Lea, M. (2003). *Writing at University, a guide for students (2nd edition)*, Maidenhead, Open University.
- Cressie, N. (1991). *Statistics for spatial data*, Chichester, Wiley.
- Cressie, N. (1993). *Statistics for Spatial Data (Revised edition)*, New York, Wiley.

- Cressie, N. (1996) Massive Data Sets: Problems and Possibilities, with Application to Environmental Monitoring, *Massive data sets: proceedings of a workshop*, 115-119
- Croner, C., Sperling, J. and Broome, F. (1996) Geographic Information Systems (GIS), *New perspectives in understanding human health and environmental relationships*, **15**(18), 961–1977
- Crookston, N. (1997). Suppose: an interface to the forest vegetation simulator. In: Richard, T., Melinda, M., Judy, A. (Ed.), *Proceedings: Forest Vegetation Simulator Conference*, pp. 3 –7.
- Dahl, J., Mehta, R. and Hoare, D. (1972). New Obligate Methyloph, *Journal of Bacteriology*, **109**(2), 916-921.
- Datta, S., Bhaduri, K., Giannella, C., Kargupta, H. and Wolff, R. (2006) Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, **10**, 18-26.
- Davis, A., Bersoff, E. and Comer, E. (1988). A strategy for comparing alternative software development life cycle models, *IEEE Transactions on Software Engineering*, **14**(10), 1453-1461.
- Day, R. (1988). *How to Write and Publish a Scientific Paper*, (3rd edition), Cambridge, Cambridge University Press.
- de Smith, M. J., M. F. Goodchild, and P. A. Longley (2007). *Geospatial Analysis: A comprehensive guide to principles, techniques, and software tools*, 2nd Edition. London, Troubador
- Decrem, B. (2006). "Introducing Flock Beta 1", Flock official blog. Available on-line at <http://www.flock.com/node/4500>, accessed 2007-01-13.
- Dehousse, S., Faulkner, S., Giorgini, P. and Kolp, M. (2005). Delegation Mechanisms for Agent Architectural Design. *Proceedings of the IEEE Intelligent Agent Technology (IAT'05)*, France, IEEE Computer Society Press, pp. 19-22.

- DeLoach, S., Padgham, L., Perini, A., Susi, A. and Thangarajah, J. (2009). Using three AOSE toolkits to develop a sample design, *International Journal of Agent-Oriented Software Engineering*, **3**(4), 416-476
- Denscombe, M. (1998). *The Good Research Guide for small-scale social research projects*, Buckingham, Open University Press.
- DePaul, F. and Sheih, C. (1985). A tracer study of dispersion in an urban street canyon, *Atmospheric Environment*, **19**, 555–559.
- Deutsch, C. (2002). *Geostatistical Reservoir Modeling*, Oxford, Oxford University Press.
- Deutsch, C. and Journel, A. (1998). *GSLIB: Geostatistical Software Library and User's Guide (2nd edition)*, New York, Oxford University Press.
- Deutsch, C.V. and Journel, A.G. (1992) *GSLIB: Geostatistical Software Library and User's Guide*. Oxford, Oxford University Press.
- Dibble, C. and Feldman, P.(2004). The GeoGraph 3D Computational Laboratory: Network and Terrain Landscapes for RePast, *Journal of Artificial Societies and Social Simulation*, **7**, 1.
- Diggle, P. (1975). Robust density estimation using distance methods, *Biometrika*, **62**, 39-48.
- Diggle, P. (1983). *Statistical Analysis of Spatial Point Patterns: Mathematics in Biology*, London, Academic Press.
- Dijkstra, A. (1994). *A computer model for bimodal sublexical processing*. In: Maarse, Akkerman, Brand, Mulder & van der Stelt (Eds.), *Computers in psychology*, Lisse, Swets & Zeitlinger, pp. 90-101.
- Dillenbourg P. (1999) What do you mean by collaborative learning?. In P. Dillenbourg (Ed) *Collaborative-learning: Cognitive and Computational Approaches*. (pp.1-19). Oxford: Elsevier

Ding, Q. (2007). Data Mining on Spatial Data. Available on-line at <http://math.nist.gov/mcsd/Seminars/2003/2003-02-25-ding.html>, accessed on 13/08/2007.

Dodge, M. and Kitchin, R. (2001). *Mapping Cyberspace*, New York, Routledge.

Dodge, M. and Kitchin, R. (2003). The role and value of maps of Internet infrastructure. In: *Moving People, Goods and Information in the Twenty First Century: Urban Technology, the New Economy, and Cutting-Edge Infrastructure*, New York, Spon Press, pp. 159-185.

Dodge, M. and Kitchin, R. (2004). Flying through code/space: The real virtuality of air travel, *Environment and Planning A*, **36**(2), 195 – 211.

Dorenbos, R. and Ramalho, J. (1998). Underbalanced Drilling Primer, *Rijswijk: Shell International Exploration and Production B.V.*

Dragicevic, S. and Balram, S. (2004) Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks, *Cartography and Geographic Information Science*, **31**, 2004

Draper, N. and Smith, H. (1981). *Applied regression analysis*, New York, Wiley.

DTL (2001). Learning through understanding, for all your IWCF requirements: Well control for the Drilling team, *Weatherford, Downhole technology Ltd.*

Dungan, L., Perry, J., Dale, M., Legendre, P., Citron-Pousty, S., Fortin, M., Jakomulska, A., Miriti, M. and Rosenberg, M. (2002). A balanced view of scale in spatial statistical analysis, *Ecography*, **25**, 626–640.

Ely, M., Vinz, R., Downing, M. and Anzu, M. (1999). *On writing qualitative research: Living by words*, London, Falmer Press.

Ester M., Kriegel H.-P., Sander J. and Xu X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, AAAI Press, pp. 226-231.

- Ester, M.; Kriegel, H.-P. & Sander, J. (2001). Algorithms and Applications for Spatial Data Mining, in Harvey J. Miller & Jiarwei Han, ed., 'Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS', Taylor and Francis,, pp. 160-187.
- Expo (1999). Well test operational guidelines, Aberdeen: The Surface & Environmental Systems Business Stream, *Expro Group*.
- Farjami, P., Goërg, C. and Bell, F. (2000). Advanced service provisioning based on mobile agents, *Computer Communications*, **23**, 754–760.
- Faulkner, B. (1999). *Java classes for nonprocedural variogram modelling*, Tarrytown, Pergamon Press, Inc.
- Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996) Knowledge Discovery and Data Mining: Towards a Unifying Framework, *KDD-96 Proceedings*, **AAAI(96)**, 82-88.
- Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). "Knowledge Discovery and Data Mining: Towards a Unifying Framework", *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, AAAI Press, pp. 82 - 88.
- Fedra, K. (1996). Distributed Models and Embedded GIS: Strategies and Case Studies of Integration. In: M. Goodchild, L. Steyart, B. Parks, C. Johnston, D. Maidment, M. Crane and S. Glendinning (eds), *GIS and Environmental Modeling: Progress and Research Issues*, Fort Collins, GIS World Books, pp.413–417.
- Feng, S., Li, L. and Cen, L. (2001). An object-oriented design tool to aid the design of manufacturing system, *Knowledge-based systems*, **14**, 225 -232.
- Fink, A. (1998). *Conducting research literature reviews: from paper to the internet*, Thousand Oaks, Sage.
- Fonseca, F. (2000). Users, Ontologies and Information Sharing in Urban GIS. *ASPRS Annual Conference*, Washington, D.C.
- Fonseca, F. and Egenhofer, M. (1999). Ontology Driven Geographical Information

Systems, *Transactions in GIS* **3**,31-49.

Fonseca, F. and Egenhofer, M. (1999). Ontology-Driven Geographic Information Systems. In: C. B. Medeiros, (eds.), *7th, ACM Symposium on Advances in Geographic Information Systems*, Kansas City, pp. 14-19.

Fonseca, F., Egenhofer, M., Davis, C. and Borges, K. (2000). Ontologies and Knowledge Sharing in Urban GIS Computer, *Environment and Urban Systems*, **24**(3), 251-272.

Fonseca, F., Egenhofer, M., Davis, C. and Câmara, G. (2002). Semantic Granularity in Ontology-Driven Geographic Information Systems. *AMAI Annals of Mathematics and Artificial Intelligence - Special Issue on Spatial and Temporal Granularity*, **36**(1-2), 121-151.

Fonseca, F., Egenhofer, M., Davis, C., and Borges, K. (2000). Ontologies and Knowledge Sharing in Urban GIS. CEUS, *Computer, Environment and Urban Systems*, **24**(3), 232-251.

Fotheringham, A., Brunson, C., and Charlton, M. (2002). *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*, Chichester, Wiley.

Frank, A. and Raubal, M.(2001). GIS Education Today: From GI Science to GI Engineering, *Journal of the Urban and Regional Information Systems Association (URISA)*, **13** (2), 5-10.

Franklin S and Graesser A (1996). Is it an agent, or just a program? A taxonomy for autonomous agents, *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Budapest, Hungary.

Friis-Christensen, A., Ostländer, N., Lutz, M. and Bernard, L. (2007). Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions, *Transactions in GIS*, **11**(6), 799–818

Gahegan, M. (1999) What is Geocomputation? *Transactions in GIS* **3** 203-206.

Galliers, R. and Land, F. (1987). *Choosing appropriate information system research methodology*, New York, ACM Press and Reading.

- Galton, F. (1886). Regression Towards Mediocrity in Hereditary Stature, *Journal of the Anthropological Institute*, **15**, 246-263.
- Garawski, M. and Pluciennik, E. (2006). Spatial Telemetric Data Warehouse and Software Agents as Environment to Distributed Execute SQL Queries, *Proceedings of the International Multiconference on Computer Science and Technology*, 243 – 253.
- Garcia-Ojeda, J., Arenas, A. and Perez-Alcazar, J. (2005). Paving the Way for Implementing Multiagent Systems: Refining Gaia with AUML, *Proceedings of International Workshop on Agent-Oriented Software Engineering (AOSE 2005)*, Utrecht, Netherlands, *Agent-Oriented Software Engineering, Lecture Notes in Computer Science*, Springer Verlag.
- Gardan, N., Gardan, Y. (2003). An application of knowledge based modelling using scripts, *Expert systems with Applications*, **25**, 555–568.
- Gardner, M. (1970). "Mathematical Games - The fantastic combinations of John Conway's new solitaire game "life"", *Scientific American*, **223**, 120–123.
- Gardner, M. (1970). Mathematical Games, The fantastic combinations of John Conway's new solitaire game "life", *Scientific American*, **223**, 120-123.
- Garzetti, M., Giorgini, P., Mylopoulos, J., Sannicolò, F. (2002). Applying Tropos Methodology to a real case study: Complexity and Criticality analysis, *Italian workshop on "Dagli OGGETTI agli AGENTI - Dall'informazione alla Conoscenza (WOA02)"*, Milano.
- GeostatisticalAnalst(2009)
<http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/0031000000nz000000>
 00 accessed on 10-01-2009
- Geovariances (2009) <http://www.geovariances.com/en/isatis-detailed-features-automatic-variogram-fitting-ar1003?lang=en> accessed on the 12-01-2009
- Getis, A. (2008) A history of the concept of spatial autocorrelation: a geographer's perspective. *Geographical Analysis* **40**, 297-309.

- Gilardi, N. (2002). "Machine learning for spatial data analysis". *A thesis for PhD, University of Lausanne and Dalle Molle Institute of Perceptual Artificial Intelligence, Matigny.*
- Gilbert, N. and Bankes, S. C. (2002) Platforms and methods for agent-based modelling. *Proceedings of the National Sciences of the United State of Amerimca*, **99**, 7197-98
- Giorgini, P. and Henderson-Sellers, B. (2005). *Agent-Oriented Methodologies: an Introduction*, Idea Group Inc.
- Giorgini. P., Bresciani, P., Giunchiglia, F., Mylopoulos, J. and Perini, A. (2001). Agent-Oriented Software Development: A Case Study, *Thirteenth International Conference on Software Engineering* .
- Giunchiglia, F., Mylopoulos, J. and Perini, A. (2002). The Tropos Software Development Methodology: Processes, Models and Diagrams, *AAMAS Conference*.
- Giupponi, C., Mysiak, J., Fassio, A. and Cogan, V. (2004). MULINODSS: A computer tool for sustainable use of water resources at the catchment scale. *Mathematics and Computers in Simulation*, **64**, 13–24.
- Goodchild, M. (1992). Geographical information science, *International Journal of Geographical Information Systems*, **6**(1), 31-45.
- Goodchild, M. (1993). *Environmental modeling with GIS*, Oxford, Oxford University Press.
- Goodchild, M. (1997). "What is Geographic Information Science?", Extracts of NCGIA Core Curriculum in Geographic Information Science. Available on-line at <http://www.ncgia.ucsb.edu/giscc/units/u002/>, accessed on 12-01-2006
- Goodchild, M. (2003). The fundamental laws of GIScience, *Keynote address of the Annual Assembly*, University Consortium for Geographic Information Science, Monterey, Carlifornia.
- Goodchild, M. (2004). The Validity and Usefulness of Laws in Geographic Information Science and Geography, *Annals of the Association of American Geographers*, **94**, 2, 300 – 303.

- Goodchild, M. (2006). GIS and disasters: Planning for catastrophe, *Computers, Environment and Urban Systems*, **30**(3), 227-229.
- Goodchild, M. (2008) Geographic Information Science: the grand challenges. In Wilson, J.P. and Fotheringham, S. (eds.) *The Handbook of Geographic Information Science*, Malden MA, Blackwell: 598-608
- Goodchild, M. F. (2006). 'GIS and disasters: Planning for catastrophe.', *Computers, Environment and Urban Systems* **30** (3) , 227-229 .
- Goodchild, M. F., Anselin, L., Appelbaum, R., and Harthorn, B. (2000). Toward spatially integrated social science, *International Regional Science Review*, **23**(2),139–159.
- Goodchild, M., Johnston, D., Maguire, D., and Noronha, V. (2004). *Distributed and mobile computing*. In: R. McMaster and E. Usery, (eds), *A Research Agenda for Geographic Information Science*, Boca Raton, CRC Press, pp. 257–286.
- Goodchild, M., Kyriakidis, P., Schneider, P., and Sifuentes, J. (2005). *Uncertainty and interoperability: the areal interpolation problem*. In: L. Wu, W. Shi, Y. Fang, and Q. Tong, (eds), *Proceedings of the Fourth International Symposium on Spatial Data Quality (ISSDQ 05)*, Beijing, 25–26-08-2005.
- Goodchild, M., Steyaert, L., Parks, B. and Johnston, C. (1996). GIS and Environmental Modeling, *Progress and Research Issues*, Wiley.
- Goovaerts, P. (1997). *Geostatistics for Natural Resources Evaluation*, New York, Oxford University Press.
- Goovaerts, P. (2006). Geostatistical analysis of disease data: accounting for spatial support and population density in the isopleth mapping of cancer mortality risk using area-to-point Poisson kriging, *International Journal of Health Geographics*, **5**,52
- Gordon, D, Billa, R, Weissman, M and Hou, F, (2003). Underbalanced Drilling with Casing Evolution in the South Texas Vicksburg, *Shell International and E&P Incorporated: prepared for the 2003 SPE Annual Technical Conference and Exhibition*, Denver.

- Gosselin, P., Lebel, G., Rivest, S. and Douville-Fradet, M. (2005). The Integrated System for Public Health Monitoring of West Nile Virus (ISPHM-WNV): a real-time GIS for surveillance and decision-making, *International Journal of Health Geography*, **4**, 21.
- Gottfredson, L. (1997). Intelligence and social policy. *Intelligence*, **24**(1).
- Graham, P. (2005). "Web 2.0: I first heard the phrase 'Web 2.0'", In the name of the Web 2.0 Conference in 2004. Available on-line at <http://www.paulgraham.com/web20.html>, accessed on 02-08-2005.
- Greenmeier, L. and Gaudin, S (2004). "Amid The Rush To Web 2.0, Some Words Of Warning -- Web 2.0 -- InformationWeek". Available on-line at www.informationweek.com, accessed 04-04-2008.
- Griffith, A. (1999). *A casebook for spatial statistical data analysis : a compilation of analyses*, Oxford, Oxford University Press
- Groff, E. (2007). 'Situating' Simulation to Model Human Spatio-Temporal Interactions: An Example Using Crime Events, *Transactions in GIS*, **11**(4), 507–530.
- Grossklags, J. and Schmidt, C. (2006). Software agents and market (in) efficiency: a human trader experiment, *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, **36**(1), 56 – 67.
- Grubbs, F. (1969). Procedures for Detecting Outlying Observations in Samples, *Technometrics*, **11**(1), 1-21.
- Gruber, T. (1991). The role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases, *Principles of Knowledge Representation and Reasoning*, Cambridge, Massachusetts, 601 – 602.
- Gruber, T. (1992). Ontolingua: A mechanism to support portable ontologies. *Stanford University, Knowledge Systems Laboratory, Technical Report no. KSL-91-66*.
- Gruber, T. (1993). A translation approach to portable ontology specifications, *Knowledge Acquisition*, **5**(2), 199–220.

- Gruber, T. (2002). What is an ontology?. Available on-line at <http://www-ksl.stanford.edu/kst/what-is-anontology.html>, accessed on 26/07/2002
- Gruber, T. (2004). What is an ontology?. Available on-line at <http://www-ksl.stanford.edu/kst/what-is-anontology.html>, accessed on 26/07/2002.
- Gruber, T. (1995). *Toward principles for the design of ontologies used for knowledge sharing*. In: N. Guarino, and R. Poli, (eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation*.
- Guarino, N. (1998). *Formal ontology and information systems*. In N. Guarino (ed.), *1st International conference on formal ontology in information systems*, Trento, IOS Press, pp. 3–15.
- Guba, E. (1961). Elements of a Proposal, *UCEA meeting in Chapel Hill, N.C.*
- Guesgen, H. and Albrecht, J. (2000). Imprecise reasoning in Geographic Information Systems. *Fuzzy Sets and Systems, Elsevier Science*, **113**, 121-131.
- Gumprecht, B. 1999. *The Los Angeles River*. Johns Hopkins University Press. Baltimore, MA.
- Gunutsky, D and Zanutto, B, (2004). Cooperation in the iterated Prisoner's dilemma is learned by Operant Conditioning Mechanism, *Artificial life*, **10**(4), 433-462.
- Haag (2005). http://en.wikipedia.org/wiki/Intelligent_agent#endnote_Haag, accessed on 11-03-2005.
- Haag, S. and Cummings, M. (2007) (6 edition). *Management Information Systems for the Information Age*, Scarborough, Irwin Professional Press.
- Haining, R. P. (2009) Spatial autocorrelation and the quantitative revolution. *Geographical Analysis* **41**, 364-374
- Halliburton (2004). The Press Releases: Halliburton wins six e&p meritorious engineering innovation awards. Halliburton

- Hamada, K., Baba, T., Sato, K., and Yufu, M. (1995). Hybridizing a genetic algorithm with rule-based reasoning for production planning, *IEEE Expert*, **35**, 60–67.
- Harrison, C., Chess, D., Kershenbaum, A. (2005). Mobile Agents: Are they a good idea? *New York, IBM Research Division, T. J. Watson Research Center*.
- Hart, C. (2003). *Doing Literature review*. London, Sage.
- Hartkamp, A., de Beurs, K., Stein, A. and White, J. (1999). Interpolation techniques for climate variables, *International Maize and Wheat Consortium, Geographic Information Systems*, series 99-0.
- Hengl, T., Grohmann, C., Bivand, R., Conrad, O. and Lobo, A. (2009). SAGA vs GRASS: A Comparative Analysis of the Two Open Source Desktop GIS for the Automated Analysis of Elevation Data, *Proceedings of Geomorphometry 2009*. Zurich, Switzerland, 31 August - 2 September, 2009
- Heppenstall, A., Evans, A. and Birkin, M. (2005). A Hybrid Multi-Agent/Spatial Interaction Model System for Petrol Price Setting, *Transactions in GIS*, **9**(1), 35-51.
- Herwin, J. (1996). A simple general model for echelon overhaul and repair, *Reliability Engineering and System*, **51**, 283 – 293.
- Herzfeld, U. and Higginson, C. (1996). Automated geostatistical seafloor classification - principles, parameters, feature vectors and discrimination criteria, *Computers and Geosciences*, **22**, 35-52.
- Hewitt, C. (1969). *Planner: A language for proving theorems in robots*. In: D. Walker, and Norton, L. (eds.), *Proceedings of the International Joint Conference on Artificial Intelligence*, May 7-9, 1969, Washington, pp. 295-301.
- Hewitt, C. (1977). Viewing Control Structures as Patterns of Passing Messages, *Journal of Artificial Intelligence*, **8**(3), 323-364
- Hewitt, C. (1977). Viewing Control Structures as Patterns of Passing Messages, *Journal of Artificial Intelligence*, **8**(3), 323-364

- Hill, M., Braaten, R., Veitch, S., Lees, B. and Sharma, S. (2004). Multi-criteria decision analysis in spatial decision support: the ASSESS analytic hierarchy process and the role of quantitative methods and spatially explicit analysis, *Environmental Modelling & Software*, **20**, 955-976.
- Hinchcliffe, D. (2006). "The State of Web 2.0". Web Services Journal. Available on-line at http://web2.socialcomputingjournal.com/the_state_of_web_20.htm, accessed 06-08-2006.
- Hodges, A. (1983). *Alan Turing: the Enigma*, London, Burnett.
- Högg, R. Meckel, M., Stanoevska-Slabeva, K., Martignoni, R. (2006). Overview of business models for Web 2.0 communities, *Proceedings of GeNeMe*, 23-37.
- Hopkins, L. (1977). Methods for generating land suitability maps: a comparative evaluation. *Journal of the American Institute of Planners*, **October edition**, 386–400.
- Hubert, L. Golledge, R. and Costanza, C. (1981). Generalized Procedures for Evaluating Spatial Autocorrelation. *Geographical Analysis*, **13**, 224–32.
- Hussain, J. and Burden, P. (1996). The Role of GIS in Managing Water Resources, *IIR Conference*, Dubai, 2-4-04-1996.
- Huynh, T., Jennings, N. and Shadbolt, N. (2006). An integrated trust and reputation model for open multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, **13** (2), 119-154.
- IBM (1996). "Subject-oriented programming: overview of concepts", Research. Available on-line at <http://www.research.ibm.com/sop/sopoverv.htm>, accessed 03-02-2005.
- IBM, (2008). The Consumer Driven Supply Chain. Retrieved on Nov 20, 2008 from <http://www-03.ibm.com/solutions/businesssolutions/sensors/doc/content/resource/thought/1529260129.html> accessed 23-12-2008

- Isaaks, E. and Srivastava, R. (1989). *An Introduction to Applied Geostatistics*, Oxford, Oxford University Press.
- JACK (2005). http://www.agent-software.com/shared/demosNdocs/practicals/jack_jde/html/p1n.html, accessed on 8-11-2005.
- Jamshidi, M., Titli, A., Zadeh, L. and Boverie, S. (1997). *Applications of fuzzy logic: Towards high machine intelligent quotient systems*, New Jersey, Prentice Hall.
- Jankowska, A., Kurbel, K., & Schreber, D. (2007). An architecture for agent-based mobile supply chain event management, *International Journal of Mobile Communications*, 5(3), 243-258.
- Jennings, N. and Wooldridge, M. (2000). *Agent-oriented software engineering, Handbook of agent technology*, Massachusetts, AAAI Press / MIT Press.
- Jiang, B., and Yao, X. (2005). Location-based services and GIS Perspective. CEUS Computer, *Environment and Urban Systems*.
- Jo, C. and Einhorn, J. (2005). A BDI Agent-Based Software Process, *Journal of Object Technology*, 4(9), 101-121.
- Johnston, K., Ver Hoef, J.M., Krivoruchko, K., Lucas, N. (2001) *The ArcGIS Geostatistical Analyst*. Redlands CA, ESRI.
- Journel, A. (1981). Non-parametric estimation of spatial distributions, *Mathematical Geology*, 15(3), 445-468
- Journel, A. and Huijbregts, C. (1978). *Mining Geostatistics*, London, Academic Press.
- JPL (1999). "Researchers stage largest military simulation ever" (news), Jet Propulsion Laboratory, Caltech, December 1997. Available on-line at, <http://www.jpl.nasa.gov/releases/97/military.html>, accessed 13-01-1999.

- Kardas, G., Goknil, A., Dikenelli, O. and Topaloglu, N. (2009). Model driven development of semantic web enabled Multi-Agent Systems. *International Journal of Cooperative Information Systems* **18**(02), 261-308
- Karimi, H., and Houston, B. (1996) Evaluating strategies for integrating environmental models with GIS: Current trends and future needs, *Computers, Environment and Urban Systems*, **20**(6), 413-425.
- Kaye, D. (2003) *Loosely Coupled: The Missing Pieces of Web Services*, California, RDS Press.
- Kent, M. and Coker, P. 1998. *Vegetation description and analysis*. London, John Wiley and Sons.
- Ki-Joune Li, K. (2007). "Ubiquitous GIS Part I: Basic Concepts of Ubiquitous GIS". Available on-line at <http://isel.cs.pnu.edu/~lik>, accessed 01-07-2006.
- King, J. (1995). Intelligent Agents: Bringing Good Things to Life, *AI Expert*, **February**, 17-19.
- Kirkby, M. (2000) Limits to modelling in the earth and environmental sciences. In A. Openshaw, and R. Abrahart (Eds), *GeoComputation*, London, Tayler and Francis, pp365-378
- Kitanidis, P. (1993). Generalised covariance functions in estimation, *Mathematical Geology*, **25**, 525-540.
- Kitanidis, P. (1997). *Introduction to Geostatistics: Applications to hydrogeology*, Cambridge, Cambridge University Press.
- Knudsen, H. (1988). Hand book for a short course on "Geostatistical Ore Reserve Estimation", Division of Mining and Minerals Engineering, Montana Tech, the University of Montana.
- Kohler, T. (2000). *Dynamics in human and primate societies – agent-based modelling of social and spatial processes*. Santa Fe Institute, *Studies in the Sciences of Complexity*, Oxford, Oxford University Press.

- Kolonder, J. (1994). *Case-based reasoning*. London, Morgan Kaufmann.
- Kriegel, H., Pötke, M. and Seidl, T. (2000) Managing Intervals Efficiently in Object-Relational Databases, *26th Proceedings International Conference on Very Large Data Bases (VLDB'00)*, Cairo, **26**, 407-418.
- Krige, D. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand, *Journal of the Chemical, Metal and Mining Society of South Africa*, **52** (6), 119–139.
- Krige, D. (1978). Lognormal DeWijsian Geostatistics for Ore Evaluation, *Geostatistics Series, South African IMM Monographs*, **1**, 55.
- Krutisch, R. and Meier, P. and Wirsing, M. (2003). The AgentComponent Approach, Combining Agents, and Components, *Multiagent System Technologies, First German Conference*, 1-12.
- Kwon, P. and Lee, J. (2001). A multi-agent intelligent system for efficient ERP maintenance, *Expert systems with application*, **21**, 191-202.
- Landim, P. (2004). INTRODUÇÃO À ANÁLISE VARIOGRÁFICA com o Variowin: Lab. Geomatemática, *DGA,IGCE,UNESP/Rio Claro, Texto Didático*, **14**, 25
- Lanter, D. and Veregin, H. (1992). A research paradigm for propagating error in layer-based GIS, *Photo-grammetric and remote sensing*, **58**, 825-833.
- Laube, P. and Duckham, M. (2009). Decentralised spatial mining for geosensor networks. In H.J. Miller and J. Han (eds.), *Geographic Data Mining and Knowledge Discovery*, Boca Raton, CRC Press, pp.409-430.
- Lawson, C. (1977) *Software for C~ surface interpolation," in Mathematical Software III*, New York, Academic Press.
- Lee, I. and Park, H. (1994). Parameterization of the pollutant transport and dispersion in urban street canyons, *Atmospheric Environment*, **28**, 2343–2349.

- Lee, J., Wing, D. and Wong, S. (2001). *Statistical analysis with ArcView GIS*, Chichester, John Wiley
- Lee, K., Yun, J. and Jo, G. (2003). MoCAAS: auction agent system using a collaborative mobile agent in electronic commerce, *Expert Systems with Applications*, **24**, 183 – 187.
- Lee, W., Liu, C. and Lu, C. (2002). Intelligent based systems for personalised recommendations in Internet Commerce, *Expert systems with application*, **22**, 275 – 284.
- Leenaers, H., Okx, J. and Burrough, P. (1990). Comparison of spatial prediction methods for mapping floodplain soil pollution, *Catena*, **17**, 535-550.
- Lefever, J., Cluff, B., Lawyer, G. and UpChurch, W. (2000). "Horizontal drilling in oil shales. Mississippi", Petroleum Technology Transfer Council web. Available on-line at <http://www.pttc.org/index.html>, accessed 28-7-2004.
- Leitl, B. and Meroney, R. (1997). Car exhaust dispersion in a street canyon. Numerical critique of a wind tunnel experiment, *Journal of Wind Engineering and Industrial Aerodynamics*, **67**, 293–304.
- Levine, N. (2007). Crime Travel Demand and Bank Robberies: Using CrimeStat III to Model Bank Robbery Trips, *Journal of social science, computer review*, **25**(2), 239-258.
- Li, W., Tsai, Y. and Chiu, C. (2004). The experimental study of the expert system for diagnosing unbalance by ANN and acoustic signals, *Journal of Sound and Vibration*, **272**, 69–83.
- Li, Y. (2005) Agent technologies for spatial data auality analysis in environmental modelling, *Proceedings GeoComputation 05*, Ann Arbor, Michigan (CD)
- Li, Y. (2006) Spatial data quality analysis with Agent technologies, *Proceedings GISRUUK2006*, Nottingham: 250-254

- Li, Y., Brimicombe, A. and Li, C. (2008). Agent-Based Services for Validating Multi-Agent Models" *Computers, Environment & Urban Systems*, **32**, 464-473.
- Li, Y., Brimicombe, A., and Ralphs, M.. (2000). Spatial data quality and sensitivity analysis in GIS and environmental modelling: the case of coastal oil spills, *Computers, Environment and Urban Systems*, **24**, 95-108.
- Liao, E., Scerri, P. and Sycara, K. (2004). A framework for very large teams. *AAMAS'04 Workshop on Coalitions and Teams*.
- Liao, S. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004, *Expert Systems with Applications*, **28**, 93–103.
- Lieberman, H., Rosenzweig, E. and Singh, P. (2001) Aria: an agent for annotating and retrieving images, *IEEE: Computer* , **34**(7), 57 – 62.
- Lieberman, H., Rosenzweig, E., Singh, P. and Wood, M. (2006) Agents for Integrated Annotation and Retrieval of Images, United States Patent No. 7028253 (application No. 09/685112), Eastman Kodak Company, New York.
- Lima, H. (2002). Experimental Validation of Well Control Procedures in Deepwater, Department of Interior, USA.
- Lin, L., Miller, H., Tsai, H. and Xu, J. (2001). Integrating a Heterogeneous Distributed Data Environment with a Database Specific Ontology. *The International Conference on Parallel and Distributed Computing Systems*.
- Lin, W., Kuo, D. and Chen, L. (2006). Global versus local sourcing for different supply chain networks: an analysis of order unfulfilment rates, *International Journal of Services Technology and Management*, **7**(5-6), 420-438
- Lloyd, C. and Atkinson, P. (1998). Scale and the spatial structure of landform: optimising sampling strategies with geostatistics. *Proceedings of the 3rd International Conference on GeoComputation*, University of Bristol, GeoComputation CD-ROM, Manchester.

- Longley, P. and Batty, M. (2003). *Prologue: Advanced Spatial Analysis: Extending GIS*. In: P. Longley and M. Batty (eds.), *Advanced Spatial Analysis: The CASA Book of GIS*, Redlands, ERSI Press, pp. 1.
- Longley, P., Goodchild, M., Maguire, D. and Rhind, D. (2002). *Geographical Information Systems and Science*, Chichester, Wiley & Sons.
- Longley, P., Goodchild, M., Maguire, D. and Rhind, D. (2005). *Geographical information systems : principles, techniques, management and applications : abridged*. Hoboken, Wiley & Sons.
- Longley, Paul A., Michael Goodchild, David J. Maguire, David W. Rhind (1998). *Geographical Information Systems: Principles, Techniques, Management and Applications*, Cambridge, GeoInformation International.
- Lucey, T. (2002). *Quantitative techniques*, London, Continuum.
- Luck, M., McBurney, P. and Preist, C. (2003). *Agent Technology: Enabling next generation computing: a roadmap for agent based computing*, Agentlink.
- Maamar, Z., Moulin, B., Babin, G. and Bédard, Y. (1999) Software Agent-Oriented Frameworks for Global Query Processing, *Journal Intelligent Information System*, **13**(3), 235-259.
- Maes, P. (1994). Agents that reduce work and information overload, *Communications of the ACM*, **37**(7), 31–40.
- Mahmoud, M. and AL-Hammad, A. (1996). An expert system for evaluating and selection of floor finishing materials, *Expert Systems with Applications*, **10**, 281–303.
- Manoukian, E. (1986). *Modern Concepts and Theorems of Mathematical Statistics*, New York, Springer-Verlag.
- Manson, S. (2007). Challenges in evaluating models of geographical complexity. *Environmental and Planning B* **34**, 245-260

- Mark, D. (2003) Geographic information science: defing the field. In Duckham et al. (Eds), *Foundations of Geographic Information Science*, London, Taylor & Francis, pp.3-18
- Mark, D.M., 2003. Geographic information science: defining the field. In M. Duckham, M.F. Goodchild, and M.F. Worboys, editors, *Foundations of Geographic Information Science*. New York: Taylor and Francis, pp. 3-18.
- Marshall, J. (2002). Developing Internet-Based GIS Applications, *GIS India*, **11**(1), 16-19.
- Martin-Baramera, M., Sancho, J., and Sanz, F. (2000). Controlling for change agreement in the validation of medical expert systems with no gold standard: PNEUMON-IA and PENOIR revisited, *Computers and Biomedical Research*, **33**, 380–397.
- Martin, D. Openshaw, S. and Abrahart, R.J. (eds.) (2001). *GeoComputation*. In: S. Openshaw and R.J. Abrahart (eds). *Environment and Planning B: Planing and Design*, **28**(3), pp. 476-477.
- Mateas, M. (2003). Mateas response to Laurel. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge, The MIT Press.
- Matheron, G. (1962). *Traité de géostatistique appliqué*, Editions Technip.
- Matheron, G. (1963). Principles of Geostatistics, *Economic Geology*, **58**, 1246–66.
- Matheron, G. (1965). *Les Variables Regionalisees et leur Estimation*, Paris, Editions Masson.
- Matheron, G. (1971). *La theory des variables regionalisees et ses applications*, Fasc.5, Les Cahiers du Centre de Morphologie Mathématiques, Ecole Naturele Suplement des Mines, Paris.
- Matheron, G. (1973). The Intrinsic Random Functions and Their Applications, *Advances in Applied Probability*, **5**, 439-468.

- McBratney, A. and Webster, R. (1981b). The design of optimal sampling schemes for local estimation and mapping of regionalised variables II, *Program and examples. Computers and Geosciences*, **7**, 335-365.
- McBratney, A., Webster, R. and Burgess, T. (1981a). The design of optimal sampling schemes for local estimation and mapping of regionalised variables, I, *Theory and method. Computers and Geosciences*, **7**, 331-334.
- McDonald J., Cohen H. and Hightower C. (2003). New Lightweight Fluids for Underbalanced Drilling. Sponsored by the U.S. Department of Energy's Federal Energy Technology Center under contract DE-AC21-94MC31197 with Maurer Engineering Inc., 2916
- McGann, C., Berger, E., Boren, J., Chitta, S., Gerkey, B., Glaser, S., Marder-Eppstein, E., Marthi, B., Meeussen, W., Pratkanis, T. and M. (2009) Wise. *Model-based, Hierarchical Control of a Mobile Manipulation Platform. In 4th Workshop on Planning and Plan Execution for Real World Systems, ICAPS*
- McIntosh, J. and Yuan, M. (2005). A framework to enhance semantic flexibility for analysis of distributed phenomena. *International Journal of Geographic Information Science*, **19**(10), 999-1018.
- McKee, L. (1996). Building the GSDI, The Open GIS Consortium, Available on-line at <http://www.opengis.org/articles/gsdi.html>, accessed on 12/09/2006.
- McMaster, R. and Usery, L. (2004) *A Research Agenda for Geographic Information Science*. Boca Raton, CRC Press.
- Meeson, B. (2000). "Earth System Science". Available on-line at http://education.gsfc.nasa.gov/ESSSProject/ess_definition.html, accessed 12-11-2004.
- Mehdi Shajari and Ali A. Ghorbani, (2004a). Fuzzy Adaptive Survivability Tool (FAST): Application of Fuzzy-BDI Agents (poster), *Faculty of Computer Science Research Expo, June 16, 2004, UNB, Fredericton, Canada*

- Mehdi Shajari and Ali Ghorbani, (2004b). Application of Belief-Desire-Intention Agents in Intrusion Detection and Response, *Proceedings of the Second Annual Conference on Privacy, Security and Trust (PST'04)*, October 13-15, 2004, New Brunswick, Canada, pp. 181-191.
- Meroney, R., Leidl, B., Rafailidis, S. and Schatzmann, M. (1996). Study of line source characteristics for 2-D physical modeling of pollutant dispersion in street canyons, *Journal of Wind Engineering and Industrial Aerodynamics*, **62**, 37–56.
- Miller, H. (2005). Necessary space-time conditions for human interaction, *Environment and Planning B: Planning and Design*, **32**, 381 – 401.
- Miller, H. J. and Han, J. (2009) *Geographic Data Mining and Knowledge Discovery*. Boca Raton, CRC Press.
- Mockler, R., Dologite, D. and Gartenfeld, M. (2000). Talk with the experts: learning management decision-making using CAI, *Cybernetics and Systems International Journal*, **31**, 431–464.
- Mooney, D. (2004). *Essay and report writing*, London, Southbank university.
- Moraitis, P. and Spanoudakis N. (2004). Combining Gaia and JADE for Multi-Agent Systems Development, *Proceedings of fourth International Symposium "From Agent Theory to Agent Implementation" (AT2AI'04)*, Vienna, Austria.
- Moraitis, P., Petraki, E. and Spanoudakis, N. (2003a). "Providing Advanced, Personalised Infomobility Services Using Agent Technology". *Twenty-third SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI2003)*, Peterhouse College, Cambridge, UK.
- Moraitis, P., Petraki, E. and Spanoudakis, N. (2003b). *Engineering JADE Agents with the Gaia Methodology*. In: R. Kowalszyk, et al. (eds.), *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*, LNAI 2592, Springer-Verlag, pp. 77-91.
- Moran, P. (1948). "The Interpretation of Statistical Maps." *Journal of the Royal Statistical Society, series B*, **10**, 243–51.

- Moran, P. (1950). Notes on Continuous Stochastic Phenomena, *Biometrika*, **37**,17-33.
- Moreno, L., Aquilar, R., Pineiro, J., Estevez, J., Sigut, J. and Gonzalez, C. (2001). Using KADS methodology in a simulation assisted knowledge based system: application to hospital management, *Expert Systems with Applications*, **20**, 235–249.
- Moulin, B., Chaker, W. and Gancet, J. (2004). PADI-Simul: an agent-based geosimulation software supporting the design of geographic spaces, *Computers, Environment and Urban Systems*, **28**(4), 387-420.
- Moulin, B., Chaker, W., Perron, J., Pelletier, P., Hogan, J. and Gbei, E. (2003). *MAGS Project: Multi-agent GeoSimulation and Crowd Simulation*. In: W. Kuhn, M. Worboys and S. Timpf, *Proceedings of the COSIT'03, Spatial Information Theory*, Springer Verlag, pp. 151-168.
- Mouratidis, H., Kolp, M., Faulkner, S, Giorgini, P. (2005). A secure architectural description language for agent systems, *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, July 25-29, The Netherlands.
- Muetzelfeldt, R. and Duckham, M. (2005). Dynamic Spatial Modelling in the Simile Visual Modelling Environment, In Fischer, P. and Unwin, D. *Re-presenting GIS*, England, John Wiley & Sons, pp.243-256
- Mulla, D. (1988). Using geostatistics and spectral analysis to study spatial patterns in the topography of southeastern Washington State, U.S.A., *Earth Surface Processes and Landforms*, **13**, 389-405.
- Mupparapu, M., Binder, R. and Cummins, J. (2005). *Use of a wireless local area network in an orthodontic clinic*, New Jersey, The American Association of Orthodontists.
- Murray, R. (2002). *How to write a thesis*, London, Open University Press.
- Mylopoulos, J., Giorgini, P. and Kolp, M., (2002). Agent Oriented Software Development, *Proceedings of the 2nd Hellenic Conference on Artificial Intelligence (SETN-02)*.

- Najlis, R. and North, M. (2005). Repast Vector GIS Integration, *14th Annual NAACSOS Conference*, Notre Dame.
- Nakamura, Y. and Oke, T. (1988). Wind, temperature and stability conditions in an east–west oriented urban canyon, *Atmospheric Environment*, **22**, 2691–2700.
- NAPA (1997). "Geographic Information for the 21st Century", Virginia: National Academy of Public Administration. Available on-line at www.fgdc.gov/nsdi/nsdi.html accessed 27-05-2004.
- Nara, A. and Torrens, P. (2007). *Spatial and temporal analysis of pedestrian egress behavior and efficiency*. In: H. Samet, C. Shahabi and M. Schneider (eds.), *Association of Computing Machinery (ACM) Advances in Geographic Information Systems*, New York, ACM, pp. 284-287.
- Nathan, A. (2002) .NET and COM: The Complete Interoperability Guide, Indianapolis, Sams Publishing.
- Nealon, J. and Moreno, A. (2003). Agent-Based Applications in Health Care in Applications of Software Agent Technology in the Health Care Domain, In J.L. Nealon, and A. Moreno (eds) *Whitestein Series in Software Agent Technologies*, Basel, Birkhäuser Verlag, pp.3-18.
- Nienaber, R & Barnard, A (2007). A generic agent framework to support the various software project management processes, *Interdisciplinary Journal of Information, Knowledge, and Management*. **12**, 151.
- Nikolai, C. & Madey, G. R. (2009). 'Tools of the Trade: A Survey of Various Agent Based Modeling Platforms.', *Journal of Artificial Societies and Social Simulation* **12**(2).
- Noh, J., Lee, K., Kim, J., Lee, J. and Kim, S. (2000). A case based reasoning approach to cognitive map-driven tacit knowledge management, *Expert Systems with Applications*, **19**, 249–259.
- Nolan, J., Simon R. and Sood, A. (2001) *Developing an Ontology and ACL in an Agent-based GIS*, National Imagery and Mapping Agency University Research Initiative Program.

- Nord-Larsen, T. and Talbot, B. (2004). Assessment of forest-fuel resources in Denmark: Technical and economic availability, *Biomass and Bioenergy*, **27**, 97 – 109.
- North, M. and Macal, C (2007). *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*, Oxford: Oxford University Press.
- Nute, D., Potter, W., Maier, F., Wang, J., Twery, M., Rauscher, H., Knopp, P. and Thomasma, S. (2004). NED-2: an agent-based decision support system for forest ecosystem management, *Environmental Modelling & Software*, **19**, 831–843
- Nute, D., Rauscher, M., Perala, A., Zhu, G., Chang, Y. and Host, G. (1995). A toolkit approach to developing forest management advisory systems in prolog, *AI Applications*, **9**(3), 39–58.
- Nwana, H. (1996). Software Agents: An Overview, *Knowledge Engineering Review*, **11**(3), 1-40.
- O'Reilly, T. (2006). "Web 2.0 Compact Definition: Trying Again". Available on-line at http://radar.oreilly.com/archives/2006/12/web_20_compact.html accessed 20-01-2007.
- O'Grady, M. and O'Hare, G. (2004). Gulliver's Genie: agency, mobility, adaptivity, *Computers & Graphics*, **28**, 677–689
- O'Hare, G. and Jennings, N. (eds.)(1996). *Foundations of Distributed Artificial Intelligence*, New York, Wiley Interscience Publications.
- O'Reilly, T. (2005). What is Web 2.0, *Design Patterns and Business Models for the Next Generation of Software*, **30**, 2005.
- Odell, J. and Bock, C. (1998). A more complete model of relations and their implementation: Roles, *Journal of ObjectOriented Programming*, **11**,51—54.
- Odell, J. and Bock, C. (1999). Suggested UML Extensions for Agents, Response to the OMG Analysis and Design Task Force UML RTF 2.0, *Request for Information no. ad/99-12-01*, IntelliCorp, Inc.

- Odell, J., Parunak, H. and Bauer, B. (2000). Extending UML for Agents, *Proceedings of the Agent-Oriented Information Systems Workshop at the AAAI00*, pp. 3-17.
- Okamoto, S., Sycara, K. and Scerri, P. (2009). Personal Assistants for Human Organizations, in *Multi-Agent Systems - Semantics and Dynamics of Organizational Models*, IGI-Global, Hershey, Pennsylvania
- Okwangale, F. and Ogao, P. (2006) Survey of Data Mining Methods for Crime Analysis and Visualization. Proceeding for 2nd Annual International Conference on Computing and ICT Research- SREC 06:221-226.
- Olea, R. (1991). *Geostatistical glossary and multilingual dictionary*, Oxford, Oxford University Press.
- Olea, R. (1999). *Geostatistics for Engineers and Earth Scientists*, Boston, Kluwer Academic Publishers.
- Openshaw, S. (2000). GeoComputation. In: S. Openshaw and R.J. Abrahart (eds.), *GeoComputation*, Taylor & Francis, New York, pp. 1-31.
- Openshaw, S. and Abrahart, R. (2000). *GeoComputation*, London, Taylor & Francis.
- Openshaw, S. and Openshaw, C. (1997). *Artificial Intelligent in Geography*, Wiley, Chichester.
- Ord, J., and Getis, A. (2001). Testing for Local Spatial Autocorrelation in the Presence of Global Autocorrelation, *Journal of Regional Science*, **41**, 411–32.
- Ovum (1994) Intelligent agents: The new revolution in software, Ovum Report, 1994.
- Palmer-Brown, D., Tepper, J. and Powell, H. (2002). Connectionist natural language. Parsing, *Trends in Cognitive Sciences*, **6**, 10.
- Pan, J., DeSouza, G. and Kak, A. (1998). FuzzyShell: A large-scale expert system shell using fuzzy logic for uncertainty reasoning, *IEEE Transactions on Fuzzy Systems*, **6**, 563–581.

- Pang Lo, C, and Yeung, A. (2006) *Concepts and Techniques of Geographic Information Systems (2nd Edition)*, New Jersey, Prentice Hall.
- Parker, C., Burden, P. and Al-Busaidi, H. (1995). The development of Water Resources GIS Applications: Concerns and Issues, *Middle East GIS User Group Meeting in Cairo, Egypt* 19-22 January 1995.
- PDO (2004). Looking into the future. *alFahal: Petroleum Development Oman newsletter*, **309**, 9-12.
- Pebesma, E. (1999). "Variogram". Available on-line at <http://www.geog.uu.nl/gstat/manual/node20.html> accessed 20-09-2006.
- Pebesma, E. and Wesseling, C. (1998). Gstat, a program for geostatistical modelling, prediction and simulation, *Computers and Geosciences*, **24**, 17-31.
- Peng, C., Xiao, S., Nie, Z., Wang, Z., and Wang, F. (1996). Applying Bayes' theorem in medical expert systems, *IEEE Engineering in Medicine and Biology*, **May/June**, 76–79.
- Peng, Z. and Tsou, M. (2003). *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks*, London, John Wiley and Son Inc.
- Perini, A. and Susi. A. (2004). Developing Tools for Agent-Oriented Visual Modelling . Multiagent System Technologies, *Proceedings for the Second German Conference, MATES 2004*.
- Perini, A. and Susi. A. (2005). Automating Model Transformations in Agent-Oriented Modelling, *Proceedings for AOSE '05 workshop*.
- Phillip, E. and Pugh, D. (1987). *How to get a PhD*, Milton Keynes, Open University Press
- Polhill, J., Gotts, N. and Law, A. (2001). Imitative Versus Non-Imitative Strategies in a land Use Simulation, *Cybernetics and Systems*, **32**(1-2), 285-307.

- Pollock, D. (1989). Documentation of computer programs to compute and display pathlines using results from the US Geological Survey modular three-dimensional finite-difference ground-water flow model. *US Geological Survey*, 89-381.
- Poslad, S. (2009) *Ubiquitous Computing Smart Devices, Smart Environments and Smart Interaction*. London, Wiley.
- Poslad, S. (2009), *Ubiquitous computing: smart devices, environments and interactions* , John Wiley & Sons Inc .
- Potok, T. Elmore, M. and Ivezic, N. (1999). Collaborative Management Environment, In *Proceedings of the InForum'99 Conference*.
- Prasad, R., Dixit, A., Malhotra, P. and Gupta, V. (2007). *Geoinformatics in precision farming: An overview*. In: A. Singh, and Chopra (eds.), *Geoinformatics Applications in Agriculture*, New India Publishing, pp. 39 – 78.
- Pringle, D. (1980). casual modelling: the simon-block approach, *Concept and techniques in modern geography*, **27**.
- Prosper, T. (2004). *Deepwater Riser Wear Technology*, Texas, Minerals Management Service.
- Purvis, M., Cranefield, S., Bush, G., Carter, D., McKinlay, B., Nowostawski, M. and Ward, R. (2000). The NZDIS Project: an Agent-Based Distributed Information Systems Architecture, *Proceedings of the Hawai'i International Conference On System Sciences*, January 4-7, 2000, Maui, Hawaii.
- Purvis, M., Cranefield, S., Ward, R., Nowostawski, M., Carter, D. and Bush, G. (2003), A multi-agent system for the integration of distributed environmental information, *Environmental Modelling and Software*, **18**(6), 565-572.
- Quint, B. (2004). Core360 Applies Intelligent Agent to Traditional Business Data. *Information today*, Issue: October 2004
- Ramalho, J. (2002). Revision of Underbalanced Drilling Primer. Rijswijk: *Shell International Exploration and Production B.V.*

- Rauscher, H., Lloyd, F., Loftis, D. and Twery, M., (2000). A practical decision-analysis process for forest ecosystem management, *Computers and Electronics in Agriculture*, **27**, 195–226.
- REC (2001). Eliminating Topside Water Treatment for Reservoir Pressure Maintenance. Raw water, *Engineering Company Ltd.*
- Rehrl, K., Rieser, H. and Salzburg, S. (2003). Next Generation Public Travel Information Systems, *ECRIM News: European Research Consortium for Informatics and Mathematics*, **54**, 48-49
- Reitsma, F. (2004). Time and Change in GIScience Modeling (as part of a panel). *Association of American Geographers Annual Meeting*, Philadelphia.
- Reitsma, F. and Albrecht, J. (2005). Implementing a new data model for simulating processes, *International Journal of Geographical Information Systems*, **19**(10), 1073-1090
- Reitsma, F. and Albrecht, J. (2006). Modeling with the Semantic Web in the Geosciences Institute of Geography, *Online Paper Series: GEO-012*, Edinburgh University.
- Repar, J. (2005) Spatio-Temporal Ontology for Digital Geographies, In Fischer, P. and Unwin, D. *Re-presenting GIS*, England, John Wiley & Sons, pp. 199-204
- Rey, S. (2009) Show me the code: spatial analysis and open source. *Journal of Geographical Systems* **11**, 191-201
- Ribes, X. (2007). "La Web 2.0. El valor de los metadatos y de la inteligencia colectiva", Telos. Cuadernos de Comunicación, Tecnología y Sociedad. Available on-line at <http://www.campusred.net/TELOS/articuloperspectiva.asp?idarticulo=2&rev=73> accessed 08-01-2008.
- Ridley, H., Atkinson, P., Aplin, P., Muller, J. and Dowman, I. (1997). Evaluating the potential of the forthcoming commercial U.S. high-resolution satellite sensor imagery at the Ordnance Survey, *Photogrammetric Engineering and Remote Sensing*, **63**, 997-1005.

- Ripley, B. (1981). *Spatial Statistics*, New York, Wiley
- Rossi, R., Mulla, D., Journel, A. and Franz, E. (1992). Geostatistical tools for modeling and interpreting ecological spatial dependence, *Ecology Monogram*, **62**, 277-314.
- Sahli, N. and Moulin, B. (2005). Agent-Based Geo-simulation to Support Human Planning and Spatial Cognition, *MABS*, **2005**, 115-132.
- Saldana A., Stein A. and Zinck J. (1998): Spatial variability of soil properties at different scales within three terraces of the Henare River (Spain). *Catena*, **33**, 139–153.
- Sánchez, D., Isern, D. and Moreno, A. (2005), An Agent-Based Knowledge Acquisition Platform., In T. Eymann; F. Klügl; W. Lamersdorf; M. Klusch and M.N. Huhns, (eds.), 'MATES', Springer, pp. 118-129 .
- Sargent, R. (1992). *Ecology of filial cannibalism in fish: theoretical perspectives*. In: M. Elgar and B. Crespi (eds.), *Cannibalism: ecology and evolution among diverse tax*, New York, Oxford University Press, pp 38-62.
- Schmidt, A., Beigl, M. and Gellersen, H.-W. (1999). There is more to context than location, *Computer & Graphics*, **6**(21), 893–901.
- Schreiber, G., Akkermans, B. and Hoog, R. (1994). CommonKads: a comprehensive methodology for KNBS development, *IEEE Expert*, **9**(6), 28–37.
- Schulze, T. Wytzisk, A. and Raape, U. (2002). Distributed Spatio-Temporal Modelling and Simulation. In: E. Yücesan, C. Chen, J. Snowdon and J. Charnes (eds.), *Proceedings of the Winter Simulation Conference*, California, USA.
- Schuurman, N. (2005). Social Dimensions of Object Definition in GIS, In Fischer, P. and Unwin, D. *Re-presenting GIS*, England, John Wiley & Sons, pp. 27-42

- Schuurman, N. (2006). Formalization Matters: Critical GIS and Ontology Research, *Annals of the Association of American Geographers*, **96**(4), 726-739.
- Schweiger, A., Sunyaev, A., Leimeister, J., & Krcmar, H. (2007). Information systems and healthcare: Toward seamless healthcare with software agents. *Communications of AIS*, **2007**(19), 692-710.
- Sengupta, R. and Bennett, D. (2003). Agent-based modeling environment for spatial decision support, *International Journal of Geographical Information Science*, **17**(2), 157–80.
- Sengupta, R. and Sieber, R. (2007). Geospatial Agents, Agents Everywhere, *Transactions in GIS*, **11**(4), 483–506.
- Sengupta, R., Bennett, D. and Wade, G. (1996). Agent mediated links between GIS and spatial modeling software using a model definition language, *Proceedings of GIS/LIS '96. Bethesda, MD*, American Congress on Surveying and Mapping: 295–309
- Shajari, M. and Ghorbani, A. (2004). Application of Belief-Desire-Intention Agents in Intrusion Detection & Response. *Conference on Privacy, Security and Trust*, 181-191.
- Sharp, J. and Howard, K. (1998). *The management of a student research project (2nd edition)*, Cambridge, Gower.
- Shehory, O., van Harmelen, F. (2004). The complex problem of coalition formation: some relaxations, *IBM Haifa Laboratories*, Israel.
- Shekhar, S., Zhang, P., Huang, Y. and Vatsavai, R. (2003). *Trends in Spatial Data Mining*. In: H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds), *Data Mining: Next Generation Challenges and Future Directions*, AAAI/MIT Press.
- Sheremetov, L., Contreras M., Valencia C. (2004). Intelligent multi-agent support for the contingency management system, *Expert Systems with Applications*, **26**, 57–71.

- Sherwin, J. (1996). A simple general model for echelon overhaul and repair, *Reliability Engineering and System*, **51**, 283 – 293.
- Shiao, D. (2004). Mobile Agent: New Model of Intelligent Distributed Computing, Part One, *IBM China*, October, 2004
- Shih, T. (2001). Mobile agent, evolution computing, *Information sciences*, **137**, 53 – 63.
- Shoham, Y. (1993) *Agent-oriented programming Artificial Intelligence*, Philadelphia, Elsevier
- Sierhuis, M. (2007). BRAHMS: Language Specification, *Technical Memorandum: TM99-0008*, NASA Ames Research Center.
- Silva, C., Castro, J., Tedesco, P., Silva, I. (2005). Describing Agent-Oriented Design Patterns in Tropos, Proceedings of the 19th Brazilian Symposium in Software Engineering, Uberlandia, Minas Gerais, Brazil, 10 – 25.
- SimWalk (2007). <http://www.simwalk.com/index.html>, accessed 13-02-2007.
- Singer, J. (2009). The Role and Regulations for Technology in Social Work Practice and E-Therapy, In A. R. Roberts (Ed). *Social Work 2.0*, New York, U.S.A.: Oxford University Press.
- Sinton, D. (1978). *The inherent structure of information as a constraint to analysis: mapped thematic data as a case study*. In: G. Dutton (ed.), *Harvard Papers on GIS, 7:SINTON*, Graduate School of Design, Harvard University, pp. 1-17.
- Skupin, A. and Fabrikant, S. (2008) Spatialization. In Wilson, J.P. and Fotheringham, S. (eds.) *The Handbook of Geographic Information Science*, Malden MA, Blackwell: 61-79
- Smith, B., and Mark, D. (1998). Ontology and Geographic Kinds, *International Symposium on Spatial Data Handling*, Vancouver, BC, Canada.
- Smith, B., and Mark, D. (1998). Ontology and Geographic Kinds, *International Symposium on Spatial Data Handling*, Vancouver, BC, Canada.

- Soh, L., Tsatsoulis, C., Gineris, D. and Bertoia, C. (2004). ARKTOS: An intelligent system for SAR sea ice image classification, *IEEE Transactions on Geoscience and Remote Sensing*, **42**, 229–248.
- Soleimani, k. (2009). Rainfall-Runoff prediction based on Artificial neural networks, *American-Eurasianj.Agric&Environ.Sci*, **5**(6), 856-865
- Somerville, I. (2002). *Software Engineering (6th edition)*, Addison-Wesley.
- Starks, T. and Fang, J. (1982). The effect of drift on the experimental semivariogram, *Mathematical Geology*, **14**, 309-319.
- Stefansky, W. (1972), Rejecting Outliers in Factorial Designs, *Technometrics*, **14**, 469-479.
- Sterling, L. and Taveter, K. (2009) *The Art of Agent-Oriented Modeling*. The MIT
- Stevens, P. and Pooley, S. (2001). *Design: An Object-Oriented Approach with UML*, London, Wiley.
- Stojanovic, D. and Djordjevic-Kajan, S. (2001) Internet GIS Application Framework for Location-Based Services Development, *Proceedings of the 7th EC-GI & GIS Workshop*
- Student, (1914). The elimination of spurious correlation due to position in time or space, *Biometrika*, **10**, 179-181.
- Sukthankar, G., Sycara, K., Giampapa, J., Burnett, C. and Preece, A. (2009). Communications for Agent-based Human Team Support, in *Organizations in Multi-Agent, IGI-Global*, Hershey, Pennsylvania, U.S.A.
- Sutherland, J. (1999). A History of Object-Oriented Programming Languages and their Impact on Program Design and Software Development.
- Tait, N., Davison, R., Whittaker, J., Leharne, S. and Lerner, D. (2004). Borehole Optimisation System (BOS)—A GIS based risk analysis tool for optimising the use of urban groundwater, *Environmental Modelling & Software*, **19**, 1111–1124.

- Tang, C., Xu, L. and Feng, S. (2001). Agent-based Geographical Information Systems. Knowledge-based systems, **14**, 233 – 242.
- Tang, W. (2008). Simulating complex adaptive geographic systems: A geographically aware intelligent agent approach, *Cartography and Geographic Information Science*, **35**(4), 239-263.
- Tatara, E., Teymour, F. and Cinar, A. (2007). Control of Complex Distributed Systems with Distributed Intelligent Agents, *Journal of Process Control*, **17**, 415-427
- Tezuka, T., Yokota, Y., Iwaihara, M., Tanaka, K. (2004) *Extraction of cognitively-significant place names and regions from web-based physical proximity co-occurrences*. In Zhou, X., Su, S., Papazoglou, M.P., Orlowska, M.E., Jeffrey, K.G., eds.: Web Information Systems (WISE 2004). Volume 3306 of Lecture Notes in Computer Science. Springer, Berlin (2004) 113—124
- Tobler, W. (1969). A computer movie simulating urban growth in the Detroit region. Paper prepared for the meeting of the *International Geographical Union, Commission on Quantitative Methods*, Ann Arbor, Michigan, August.
- Tobler, W. (1970). A computer movie simulating urban growth in the Detroit region, *Economic Geography*, **46**, 234–40.
- Tobler, W. 1970. A computer movie simulating urban growth in the Detroit region, *Economic Geography*, **46**, 234–40.
- Tomlin, C. (1990). *Geographic Information Systems and Cartographic Modeling*, New Jersey, Prentice Hall,
- Torrens, P. (2001). New tools for simulating housing choices Program on Housing and Urban Policy, *Conference Paper Series CO1–006*, Berkeley, University of California Institute of Business and Economic Research and Fisher Center for Real Estate and Urban Economics.
- Torrens, P. (2002). SprawlSim : modeling sprawling urban growth using automata-based models Agent-based models of land-use and land-cover change :report and review

of an international workshop, October 4-7, 2001, Irvine, California, USA (2002), Lucc report series, **6**, 72-79.

Torrens, P. (2003). Simulating Sprawl: A Dynamic Entity-Based Approach to Modeling North American Suburban Sprawl Using Cellular Automata and Multi-Agent Systems.

Torrens, P. (2004a). *Geosimulation and transport modeling*. In: P. Stopher, K. Button, K. Haynes, D. Hensher (eds.), *Handbook of Transportation Geography and Spatial Systems* 5, London, Pergamon/Elsevier Science, pp. 549-565.

Torrens, P. (2004b). Emerging trends in spatial simulation, IGERT, New York: State University of New York at Buffalo.

Torrens, P. (2005). Geosimulation approaches to traffic modelling. In: P. Stopher, K. Button, K. Haynes, D. Hensher (eds.), *Transport Geography and Spatial Systems*, London, Pergamon, pp.

Torrens, P. (2006). *Urban simulation at the micro-level*, San Rafael, Autodesk, Inc.

Torrens, P. (2007a). "Modeling crowd behaviour". Available on-line at <http://www.geosimulation.org/crowds/#overview>, accessed 21-07-07.

Torrens, P. (2007b). Behavioral Intelligence for Geospatial Agents in Urban Environments. *IEEE/WIC/ACM'07*, 63 – 66.

Torrens, P. (2007c). <http://www.geosimulation.org/geosim/> accessed 23-09-2007.

Torrens, P. (2007d). A geographic automata model of residential mobility, *Environment and Planning B: Planning and Design*, **34**, 200 – 222.

Torrens, P., and O'Sullivan, D. (2001). Cellular automata and urban simulation: where do we go from here? *Environment and Planning B*, **28**(2), 163-168.

Travers, M. (1994). *Recursive Interfaces for Reactive Objects*, Massachusetts, MIT Media Laboratory.

- Travers, M. (1996). Agent-based programming, *IEEE Symposium on Visual Languages and Second Workshop on End-User Programming and Education*, Colorado, USA.
- Tripathi, P. (2002). *A text book of research methodology in social sciences*, Sultan Chan and Sons, New Delhi, Turban & Aronson.
- Troelsen, A. (2002). *COM and .NET Interoperability*, New York, Springer-Verlag.
- Trunick, P. (2007). Risks multiply with IT investments. *Logistics Today*, **48**(4), 16-16.
- Tryon, R. (1939). *Cluster Analysis*. Edwards Brothers.
- Tukey, J. (1977). *Exploratory Data Analysis*, Reading, Addison-Wesley.
- Turban, E. and Aronson, J. (2001). *Decision Support Systems and Intelligent Systems (6th edition)*, Englewood Cliffs, Prentice-Hall.
- Turing, A.M. (1940b), Letter to Newman, dated 21 April, "1940". In R.O. Gandy, *AMT D/2, Contemporary Scientific Archives Centre*, King's College Library, Cambridge Press.
- Twery, M., Rauscher, H., Bennett, D., Thomasma, S., Stout, S., Palmer, J., Hoffman, R., DeCalesta, D., Gustafson, E., Cleveland, H., Grove, J., Nute, D., Kim, G. and Kollasch, R. (2000). NED-1: integrated analyses of forest stewardship decisions, *Computers and Electronics in Agriculture*, **27**, 167–193.
- van Breemen, A. and de Vries, T. (2001). Design and implementation of a room thermostat using an agent-based approach, *Control Engineering Practice*, **9**, 233-248.
- van Vliet, J., Dragecivic, S. and White, R. (2009). 'Modelling urban growth using a variable grid cellular automaton', *Computers, Environment and Urban Systems*, **33**, 35-43.
- Vckovski, K. Brassel, and H.-J. Schek (1999). Preface. Interoperating Geographic Information Systems, *Second International Conference, INTEROP'99*, Zurich, Switzerland, pp. 5-6.

- Venables, W. and Ripley, B. (1997). *Modern Applied Statistics with S-PLUS (2nd edition)*, Oxford, Springer.
- Von Neumann, J. and Burks, A. (1966). *Theory of self-reproducing automata*, Urbana,, University of Illinois Press.
- Voyles, R. (1997). "Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation", PhD thesis, Pittsburgh, Pennsylvania, Carnegie Mellon University.
- Waddell, Paul and Shukla, V. (1993) Employment Dynamics, Spatial Restructuring and the Business Cycle, *Geographical Analysis*, **25**, 35-52.
- Walker, J. and Veitch, S. (2001). Assessment of catchment condition in Australia's intensive land use zone: a biophysical assessment at the national scale, *Final Report on Project 7/7 NLWRA to the National Land and Water Resources Audit*, November 2000, p. 74.
- Web 2.0 Conference (2007). www.conferences.oreillynet.com, accessed 08-11-2007.
- WebMashup (2007). <http://www.webmashup.com/>, accessed 08-11-2007.
- Wei, Z., Jianbang, H. and Tianhe, C. (1992). Application of expert systems in land resources research, *Computers, Environment and Urban Systems*, **16**(4), 321-327.
- Weidmann, N. and Girardin, L. (2006) GROWLab: A Toolbox for Social Simulation, *First World Congress on Social Simulation*, August 21-25, Kyoto, Japan.
- Weiss, G. (1999), *Multiagent systems: a modern approach to distributed artificial intelligence* MIT Press , Cambridge, MA, USA
- Weissberg, R. and Burker, S. (1990). *Writing up research: experimental learning*, London, Prentice Hall.
- West Jr., L. and Hess T. (2002). Metadata as a knowledge management tool: supporting intelligent agent and end user access to spatial data, *Decision Support Systems*, **32**, 247– 264

- Whigham, P. and Davis, J. (1989). Modelling with an integrated GIS/ Expert System, *Ninth Annual ESRI Users Conference*, Palm Springs, California.
- Wiederhold, G. (1994). Interoperation, Mediation and Ontologies, *International Symposium on Fifth Generation Computer Systems (FGCS94)ICOT*, Tokyo, pp. 33-48.
- Wiig, K. (1994). *Knowledge management, the central management focus for intelligent-acting organization*, Arlington, Schema Press.
- Wijngaards, N., Overeinder, B., van Steen, M. and Brazier, F. (2001). Supporting Internet-scale multi-agent systems, *Data & Knowledge Engineering*, **41**, 229–245
- Wood, J. (1996). "LandSerf: A Geographic Information System for the Visualization and Analysis of Surfaces". Available on-line at <http://www.landserf.org/>, accessed 12-07-2007.
- Wooldridge, M, Jennings N. and Kinny, D., (2000). The Gaia methodology for agent-oriented analysis and design, *Journal of Autonomous Agents and Multi-Agent Systems*, **3**(3),285–312.
- Wooldridge, M. (2000). *Reasoning about Rational Agents*, Massachusetts, The MIT Press.
- Wooldridge, M. (2002) Intelligent Agents: The Key Concepts, In V. Mark, et al. (Eds.), *MASA 2001, LNAI 2322*, Berlin, Springer-Verlag, pp. 3 – 43
- Wooldridge, M. (2003). *An Introduction to Multiagent Systems*, Chichester, John Wiley and Sons Ltd.
- Wooldridge, M. (2004). *Social Laws in Alternating Time*. In: A. Lomuscio and D. Nute (eds.), *Deontic Logic in Computer Science, 7th International Workshop on Deontic Logic in Computer Science, DEON 2004*, Madeira, Springer.
- Wooldridge, M. and Jennings, N. (1995). Intelligent agents: Theory and practice, *The Knowledge Engineering Review*, **10**(2), 115–152.

- Wooldridge, M. and Jennings, N. (1995). Intelligent agents: Theory and practice, *The Knowledge Engineering Review*, **10**(2),115–152.
- Wooldridge, M. and Jennings, N. (1999). Software engineering with agents: pitfalls and pratfalls, *IEEE Internet Computing*.
- Wooldridge, M. and Jennings,N. (1998). Pitfalls of agent-oriented development. *Proceedings of the 2nd International Conference on Autonomous Agents*, Minneapolis, pp. 385-391.
- Wooldridge, M., (2000). *Intelligent agents, Multiagent systems: a modern approach to distributed artificial intelligence*. In: G. Weiss (ed.), 2000, Massachusetts, MIT Press, pp. 27–77.
- Wooldridge, M., Jennings, N. and Kinny, D. (1999). A methodology for agent-oriented analysis and design, *Proceedings of the third international conference on Autonomous Agents (Agents-99)*, Seattle.
- Worboys, M. and Hornsby, K. (2004). *From objects to events: GEM, the geospatial event model*. In: M. Egenhofer, C. Freksa, H. Miller (eds.), *Proceeding of GIScience 2004, Lecture Notes in Computer Science*, 3234, Berlin, Springer, pp. 327-343.
- Wright, D., M. Goodchild, and J. Proctor, (1997). Demystifying the persistent ambiguity of GIS as 'tool' versus 'science', *Annals of the Association of American Geographers*, **87**(2), 346-362.
- Wu, X., Zhang, S. and Goddard, S.(2004). Development of a Component-based GIS using GRASS. *Proceedings of the FOSS/GRASS Users Conference - Bangkok, Thailand, 12-14 September 2004*
- Xia, L. and Xiaoping, L. (2006). An extended cellular automaton using case-based reasoning for simulating urban development in a large complex region, *International Journal of Geographical Information Science*, **20**(10), 1109-1136.
- Xu, R., Chu, D., Zhao, Z., Zhang, K. and Lin, P. (2008). An Architecture for Agent-Based Distributed Component Repository, *International Seminar on Business and Information Management*, **2**,116-120.

- Yang, C. and Raskin, R. (2009) Introduction to distributed geographic information processing research. *International Journal of Geographical Information Science* **23**, 553-560
- Young, R. M. (1973) *The Human Limits of Nature*. In: J. Benthall (ed.), *The Limits of Human Nature*, Allen Lane, pp. 235-74.
- Yu, B., Li, C. Sycara, K. (2009). An Incentive Mechanism for Message Relaying in Unstructured Peer to Peer Systems, *Electronic Commerce Research and Applications*, **8**(6), 315-326
- Yu, H., Shen, Z. and Miao, C. (2007) Intelligent Software Agent Design Tool Using Goal Net Methodology, *International Conference on Intelligent Agent Technology, 2007. IAT '07. IEEE/WIC/ACM* , 43 – 46.
- Yuan, M. (2001). Representing Complex Geographic Phenomena in GIS, *Cartography and Geographic Information Science*, **28**, 83 – 96.
- Zhao, W. and Jo, C. (2003). A compiler design for the agent-based programming language, *ISCA CATA-2003*.
- Zhao, W. and Jo, C. (2003). A Compiler Design for the Agent-Based Programming Language, *Computers and Their Applications*, 393-396.
- Zipf, A. and Jöst, M. (2006). Implementing adaptive mobile GI services based on ontologies: Examples from pedestrian navigation support, *Computers, Environment and Urban Systems*, **30**, 784–798.
- Zivan, R., Ginton, R., Sycara, K., (2009). Distributed Constraint Optimization for large teams of mobile sensing agents, *In proceedings of the International conference on Intelligent Agent Technology*, Milan, Italy.

APPENDIX A: AGENT FUNCTION AND ALGORITHMS

Agent Name: Data Finder			
Capability	Algorithm		
Data Finding	<p>At this point the data is searched in the approved location.</p> <p>Pre-defined locations are searched periodically to check if any new data has arrived.</p> <p>Then the actual data is scrutinised to find if it have been used (dealt) for before.</p> <p>If the data is found to have been dealt with then the agent does not waste time to re-analyse it but simple show the previous result where the expert user can say if the data is to be re-analysed and assist on new strategy.</p>	Plans and Events Associated	
		Plan	Event
		Get Data	Get Data Data Does not exist Data Arrived
Data Identifying	<p>The agents simultaneously check for clusters and repetition.</p> <p>Start by checking the repetition heuristically.</p> <p>Find the pattern if the repetition is perceived by the agent to be erratic and possible due to error then it is removed otherwise move to the next strategy.</p> <p>Check for clusters using the variance over mean formula which if returns zero means there is a clusters in the data and also using kMean cluster analysis we will pick out the clusters and visually display them to expert user.</p> <p>Heuristically check if data is divided into equal slices.</p> <p>Only do if the data has repetition (the main function here is to determine if the data is to be sliced for more analysis or not and if yes what structure to follow).</p>	Checking Repetition	Check Repetition Data Arrived Split Data
		Check Clusters	Check Repetition Data does not exist
		Determine slice	Data Arrive Split data

Agent Name: Integrity checker			
Capability	Algorithm		
Data Cleaning	<p>Separate the collected outlier data into a file and provides the two separated file one with outlier one without.</p> <p>Keep back the data (determine if the data removed is of interest or not in case the resulting analysis is not favoured by the expert and the agents needs to re-evaluate which data has to go back into the overall analysis).</p> <p>Check if data has normal distribution</p> <p>Use the Grubs technique for detecting outlier (Grubbs' test (Grubbs 1969) is mainly used to detect outliers in a univariate data set that is assumed to have normal distribution. Thus to apply this test on a dataset it is important to determine the that the data can be reasonably approximated by a normal distribution).</p> <p>Detect one outlier at a time (each data set is iterative examine using formula and checked if it is an outlier).</p> <p>Repeat iteration for all data set and again the iteration is repeated until there is no outlier detected (this multiple iterations mechanism changes the probabilities of detection and thus the test is not used for very small dataset sample sizes.</p> <p>check if the resulting analysis is deemed insufficient whether by the expert user or the inter-agent communication (other agent that failed to come up with conclusion due to missing information).</p>	Plans and Events Associated	
		Plan	Event
		Clean Data	Data cleaned Outlier found
		Find Extreme Outliers	Outlier found Data cleaned Technique suggested
		Re-examine Data	Data cleaned Data problem Data to be sampled Data arrived
Data sorting	<p>Sort the data in three structures:</p> <ol style="list-style-type: none"> 1. The x axis sorting 2. The y axis sorting 3. The actual data under study sorting 	Sort Data	Technique suggested Data arrive

Finding Distribution	<p>Determine the techniques to be used (robust, parametric or both). Check for all the required processes individually (checking clusters, trends etc) and the general preferred technique.</p> <ul style="list-style-type: none"> × Determine there is a clusters in the data and robust techniques preferred when handling the overall data × Determine that the data is regular and that parametric technique is good × Determine that the data is normal and parametric is good 	Suggest Technique	Data arrived Technique suggested
-----------------------------	--	-------------------	-------------------------------------

Agent Name: Data Analyser			
Capability	Algorithm		
Data Analysing	<p>finds if there is a trend in the data and identifies the type of trend. Formula used for finding and describing trends are those presented by Unwin and Davids:</p> <p>Here we use a polynomial regression formula. The constants are determined using matrix algorithms provided by JAMA.</p> <p>the value being observed v, x, y, residual u, the trend value z</p> <p>Check the extent of the trend and determined if it must be removed for data analysis or negligible (simply apply formula to remove trend from the data when required).</p>	Plans and Events Associated	
		Plan	Event
		Find Trend	Data to be Sampled Data to be modelled
		Remove Trend (Check)	Trend found Data to be sampled Data to be modelled Data cleaned

Agent Name: Sampler			
Capability	Algorithm		
Sampling	Provide a sample of the overall data by using well known sampling strategies. Use inbuilt strategies: <ul style="list-style-type: none"> × Stratified Random × Quasi Random × Cluster based sample 	Plans and Events Associated	
		Plan	Event
		Sample	Data to be sampled Data to be modelled

Agent Name: Mathematical Modeller			
Capability	Algorithm		
Applying formula	Determine scale Choose Lag plan the lag for the variogram plays a crucial role in producing a valid variogram analysis. Thus here using the formula provided by Isaak and srivastava (1989), p_0 - the agent determine appropriate lag and test until appropriate lag is determined. The lag is determined by getting the average spacing between dataset as starting lag (Isaak and srivastava 1989 p 146). If the data is normally distributed then the spacing between the data is chosen (Isaak and srivastava 1989 p 146). This lag is then checked if there are less than 30 values in the lag chosen for all the separation. If there are less than 30 then accordingly the lag is iteratively decreased	Plans and Events Associated	
		Plan	Event
		Determine Scale	Data to be modelled Fit model
		Choose Lag	Data to be modelled Fit model

	<p>and increased by extra 10% of the mean value. Also to give the lag values some tolerance we allow is some difficult (the data with strong trends in multi direction, to clustered data, and the data size that is too large, too small) circumstance to have the allowed values on the lag(30) + 10 and -10. However this will be determined as suspect fit.</p> <p>Plan at this point a parametric or robust variogram model is applied to the data. This is done after the data is determined if required to be checked if correlated.</p>	Apply Formula	Data processed Fit model
--	--	---------------	-----------------------------

Agent Name: Strategy Comparer			
Capability	Algorithm		
Comparing	<p>Test the hypothesis to see is the sampling strategy was good and in turn the variogram.</p> <p>Simply compare the result of the sampling data to of that of real data and try to see the offset. Use:</p> <ul style="list-style-type: none"> × Chi Square × Student test × Simply using mean, median, variance, max, min of dataset 	Plans and Events Associated	
		Plan	Event
		Compare	Send similar results Data processed

Agent Name: Model Fitter			
Capability	Algorithm		
Fitting Model	Choose appropriate model from the universal models. Here the universal models that are already known by the agent are:	Plans and Events Associated	
		Plan	Event

	<ul style="list-style-type: none"> × Spherical (valid in R^1, R^2, R^3) × Gaussian × Exponential (valid in $R^d, d>0$) × Pure Nugget (where $c \geq 0$ is nugget) × Linier (valid in $R^d, d>0$) <p>the same lag, range and sill deducted for the parametric variogram will be chosen by making sure that:</p> <ol style="list-style-type: none"> 1. the model fit lies to the right of the mathematical model otherwise change the scale (e.g.) length value. 2. If more than 50% and the rest lie to the right of the points estimated by the mathematical model lie on the same line as the model it is good 3. The range is the first dipping point of the mathematical model <p>A non-zero nugget indicated that repeated measurements at the same point yield different values. The model chosen are also determined:</p> <p>If the range is smaller than the sill spherical model is good</p> <p>If the range is almost twice than the sill gaussian model is good</p> <p>If the range is larger than the sill exponential model is good</p> <p>Model Plotter returns that the strategy used to model the variogram were not satisfactory this plan takes place to re-analyse the data. The data is tagged to what has been to it and not work so the techniques are not repeated.</p>	Fit Data to Model	<p>Validate model</p> <p>Send similar results</p> <p>Data problem</p>
		Re-Fit Data to Model	<p>Model wrong</p> <p>Validate model</p>
Validate Model	Use Ordinary Kriging to estimate a percentage of values and determine if the variogram fit was good.	Validate Goodness of Fit	<p>Model wrong</p> <p>Model fit</p>

	weights used are deducted using the weighted least square method (This weight coefficients makes the cost criteria sensible to the number of pairs and to the value of the variogram)		Validate model
	<p>validate the variograms using Student test and the Indicative Goodness of Fit (To choose the best fitting result and possibly to improve it by some small changes not any cost function can be used. They can be either the other goal functions of minimisation procedure (for example mentioned above) or some the indicators of fitting quality).</p> <p>for small distances use the Cressie method give the same results and for larger distance use Student test and/or the Indicative Goodness of Fit (In the variogram List for each lag I have to save the lag distance the min distance the max distance the number of pairs)</p> <p>Check the value of the Indicative Goodness of fit is close to zero then the fitness is good. It can be used for all directions together or for each separately.</p> <p>Usually use the Indicator of goodness of fit for one direction.</p>		

Agent Name: Model Plotter			
Capability	Algorithm		
Model Plotting	this will choose the appropriate representation scale on the Graphical User Interface. If it realises there is any issue in representing the current value it flags to have the model refitted. This agent is also responsible on communicating with the expert user who can decide if the model fitted is inappropriate and ask the agent to refit.	Plans and Events Associated	
		Plan	Event
		Plot	Model fit Model wrong

APPENDIX B: THE SAMPLE DATA (WALKER LAKE DATA)

Index for the values:

- ID Identification Number
- X location in meter
- Y location in meter
- V concentration in ppm
- U concentration in ppm
- T indicator variable

ID	X	Y	V	U	T
1	11	8	0	1.00E+31	2
2	8	30	0	1.00E+31	2
3	9	48	224.4	1.00E+31	2
4	8	69	434.4	1.00E+31	2
5	9	90	412.1	1.00E+31	2
6	10	110	587.2	1.00E+31	2
7	9	129	192.3 1	1.00E+31	2
8	11	150	31.3	1.00E+31	2
9	10	170	388.5	1.00E+31	2
10	8	188	174.6	1.00E+31	2
11	9	209	187.8	1.00E+31	2
12	10	231	82.1	1.00E+31	1
13	11	250	81.1	1.00E+31	1
14	10	269	124.3	1.00E+31	2
15	8	288	188	1.00E+31	2
16	31	11	28.7	1.00E+31	2
17	29	29	78.1	1.00E+31	2
18	28	51	292.1	1.00E+31	2
19	31	68	895.2	1.00E+31	2
20	28	88	702.6	1.00E+31	2
21	30	110	490.3	1.00E+31	2
22	28	130	136.1	1.00E+31	2
23	28	150	335	1.00E+31	2
24	30	171	277	1.00E+31	2
25	28	190	206.1	1.00E+31	2
26	31	209	24.5	1.00E+31	2
27	28	229	198.1	1.00E+31	2
28	30	250	60.3	1.00E+31	2
29	31	269	312.6	1.00E+31	2

30	31	289	240.9	1.00E+31	2
31	49	11	653.3	1.00E+31	2
32	49	29	96.4	1.00E+31	2
33	51	48	105	1.00E+31	2
34	49	68	37.8	1.00E+31	2
35	50	88	820.8	1.00E+31	2
36	51	109	450.7	1.00E+31	2
37	48	129	190.4	1.00E+31	2
38	49	151	773.3	1.00E+31	2
39	51	168	971.9	1.00E+31	2
40	48	190	762.4	1.00E+31	2
41	50	211	968.3	1.00E+31	2
42	49	231	394.7	1.00E+31	2
43	51	250	343	1.00E+31	2
44	50	268	863.8	1.00E+31	2
45	51	290	159.6	1.00E+31	1
46	71	9	445.8	1.00E+31	2
47	71	29	673.3	1.00E+31	2
48	70	51	252.6	1.00E+31	2
49	68	70	537.5	1.00E+31	2
50	69	90	0	1.00E+31	2
51	68	110	329.1	1.00E+31	2
52	68	128	646.3	1.00E+31	2
53	69	148	616.2	1.00E+31	2
54	69	169	761.3	1.00E+31	2
55	70	191	918	1.00E+31	2
56	69	208	97.4	1.00E+31	1
57	69	229	0	1.00E+31	1
58	68	250	0	1.00E+31	1
59	71	268	0	1.00E+31	1
60	71	288	2.4	1.00E+31	1
61	91	11	368.3	1.00E+31	2
62	91	29	91.6	1.00E+31	2
63	90	49	654.7	1.00E+31	2
64	91	68	645.5	1.00E+31	2
65	91	91	907.2	1.00E+31	2
66	91	111	826.3	1.00E+31	2
67	89	130	975.3	1.00E+31	2
68	88	149	551.1	1.00E+31	2
69	89	170	155.5	1.00E+31	1
70	89	188	10.7	1.00E+31	1

71	90	211	0	1.00E+31	1
72	90	230	0	1.00E+31	1
73	88	249	0	1.00E+31	1
74	88	269	12.1	1.00E+31	1
75	88	288	62.2	1.00E+31	1
76	109	11	399.6	1.00E+31	2
77	111	31	176.6	1.00E+31	2
78	108	49	402	1.00E+31	2
79	109	68	260.6	1.00E+31	2
80	108	88	192	1.00E+31	2
81	110	109	237.6	1.00E+31	2
82	109	129	702	1.00E+31	2
83	110	148	38.5	1.00E+31	2
84	111	169	22.1	1.00E+31	1
85	111	191	2.7	1.00E+31	1
86	110	208	17.9	1.00E+31	1
87	109	230	174.2	1.00E+31	2
88	109	249	12.9	1.00E+31	2
89	109	268	187.8	1.00E+31	2
90	111	291	268.8	1.00E+31	2
91	130	9	572.5	1.00E+31	2
92	131	31	29.1	1.00E+31	2
93	130	48	75.2	1.00E+31	2
94	128	70	399.9	1.00E+31	2
95	129	90	243.1	1.00E+31	2
96	131	109	0	1.00E+31	2
97	129	128	244.7	1.00E+31	2
98	131	148	185.2	1.00E+31	2
99	131	169	26	1.00E+31	1
100	129	191	0	1.00E+31	1
101	128	209	100.3	1.00E+31	1
102	130	231	530.3	1.00E+31	2
103	131	248	107.4	1.00E+31	2
104	128	269	159.3	1.00E+31	2
105	131	288	70.7	1.00E+31	2
106	148	8	260.2	1.00E+31	2
107	149	29	326	1.00E+31	2
108	150	49	332.7	1.00E+31	2
109	151	69	531.3	1.00E+31	2
110	150	89	547.2	1.00E+31	2
111	150	109	482.7	1.00E+31	2

112	150	129	84.1	1.00E+31	2
113	150	151	4.7	1.00E+31	2
114	149	169	180.6	1.00E+31	2
115	151	190	0. 1E	1.00E+31	1
116	148	208	342.4	1.00E+31	2
117	150	228	602.3	1.00E+31	2
118	149	251	209.1	1.00E+31	2
119	149	271	79.4	1.00E+31	2
120	148	291	104.1	1.00E+31	2
121	168	8	446	1.00E+31	2
122	171	29	189.9	1.00E+31	2
123	169	49	280.4	1.00E+31	2
124	168	69	0	E31 1	1
125	168	91	499.3	1.00E+31	2
126	171	109	457.3	1.00E+31	2
127	168	131	341.2	1.00E+31	2
128	171	150	0	1.00E+31	2
129	171	171	208.3	1.00E+31	2
130	169	191	99.7	1.00E+31	1
131	170	210	636.6	1.00E+31	2
132	170	230	173.1	1.00E+31	2
133	169	249	17	1.00E+31	2
134	168	271	283.1	1.00E+31	2
135	168	290	30.9	1.00E+31	1
136	190	11	348.5	1.00E+31	2
137	191	28	222.4	1.00E+31	2
138	191	48	59.1	1.00E+31	2
139	190	69	0	1.00E+31	1
140	190	89	326	1.00E+31	2
141	188	111	325.1	1.00E+31	2
142	191	129	114.7	1.00E+31	2
143	189	149	481.6	1.00E+31	2
144	190	169	324.1	1.00E+31	2
145	190	189	10.9	1.00E+31	1
146	188	210	332.9	1.00E+31	2
147	191	231	184.4	1.00E+31	2
148	190	248	146.6	1.00E+31	2
149	189	270	92	1.00E+31	1
150	189	290	2.5	1.00E+31	1
151	211	11	358.1	1.00E+31	2
152	209	30	473.3	1.00E+31	2

153	211	49	308.8	1.00E+31	2
154	210	70	406.8	1.00E+31	2
155	209	90	812.1	1.00E+31	2
156	210	111	339.7	1.00E+31	2
157	211	130	223.9	1.00E+31	2
158	208	151	673.5	1.00E+31	2
159	209	168	141	1.00E+31	2
160	208	191	61.8	1.00E+31	1
161	210	211	258.3	1.00E+31	2
162	211	228	590.3	1.00E+31	2
163	211	250	166.9	1.00E+31	2
164	208	268	125.2	1.00E+31	2
165	208	289	29.3	1.00E+31	1
166	231	10	617.6	1.00E+31	2
167	231	28	425.9	1.00E+31	2
168	230	50	295.7	1.00E+31	2
169	230	71	224.9	1.00E+31	2
170	229	91	31.7	1.00E+31	1
171	229	110	377.4	1.00E+31	2
172	230	131	333.3	1.00E+31	2
173	228	148	351	1.00E+31	2
174	229	169	0.0	1.00E+31	1
175	231	191	137.6	1.00E+31	2
176	231	208	451.2	1.00E+31	2
177	229	228	639.5	1.00E+31	2
178	231	249	119.9	1.00E+31	2
179	231	268	27.2	1.00E+31	1
180	230	291	2.1	1.00E+31	1
181	249	9	167.7	1.00E+31	1
182	250	30	147.8	1.00E+31	2
183	249	48	442.7	1.00E+31	2
184	251	69	487.7	1.00E+31	2
185	251	91	0	1.00E+31	1
186	248	109	28.2	1.00E+31	1
187	249	130	0	1.00E+31	1
188	248	150	18.3	1.00E+31	1
189	250	169	266.3	1.00E+31	2
190	250	190	502.3	1.00E+31	2
191	251	208	0	1.00E+31	2
192	251	229	240.9	1.00E+31	2
193	249	251	234.4	1.00E+31	2

194	248	270	22.4	1.00E+31	1
195	250	291	45.6	1.00E+31	1
196	40	71	76.2	1.1	2
197	21	69	284.3	7.8	2
198	28	80	606.8	105.3	2
199	29	59	772.7	1512.7	2
200	41	81	269.5	9.8	2
201	18	80	1036.7	860.4	2
202	39	60	238.6	12.7	2
203	18	60	519.4	177.1	2
204	41	90	414.9	23.4	2
205	21	90	601.4	173.1	2
206	31	101	579.2	296.5	2
207	41	100	601.4	300.6	2
208	21	100	594.6	229.7	2
209	60	8	550.1	258.3	2
210	40	11	99.4	2.2	2
211	51	18	233.6	14.2	2
212	59	20	14.4	0.1	2
213	41	21	115.9	3.1	2
214	59	90	506.2	126.9	2
215	51	101	502.4	73.8	2
216	50	81	608	210.7	2
217	59	101	363.9	30.4	2
218	60	81	385.6	50.3	2
219	60	151	1521.1	3691.8	2
220	38	148	340.9	50	2
221	50	160	879.1	474.2	2
222	50	138	413.4	83	2
223	61	158	868.9	983.8	2
224	39	160	657.4	217.8	2
225	61	139	477	71.5	2
226	38	140	268.5	26.2	2
227	61	170	806.4	301.1	2
228	39	170	914.4	1548.5	2
229	49	179	811.5	234.9	2
230	58	179	1113.6	2154.7	2
231	39	181	1008	3637.4	2
232	60	191	1528.1	1930.9	2
233	40	190	970.9	1391.1	2
234	51	198	1109	1660.8	2

235	60	198	1203.9	1813.7	2
236	40	200	641.3	249.1	2
237	58	208	720.6	1160.1	2
238	38	209	665.3	547.8	2
239	50	221	543.3	1066.6	2
240	61	220	101.1	59.5	1
241	39	221	615.9	420.9	2
242	59	268	543.1	1714.2	2
243	41	271	868.8	828.7	2
244	49	278	583	1788.8	2
245	51	260	670.7	3738.9	2
246	59	281	148.8	675	1
247	39	279	798	1182.1	2
248	59	258	194.9	983.3	1
249	38	260	635.2	766.6	2
250	78	28	781.6	565.4	2
251	60	29	238.6	12.7	2
252	70	41	472	84.9	2
253	70	21	58.1	0.3	2
254	78	41	600.3	124.6	2
255	61	41	64.9	0.8	2
256	78	20	505.9	70	2
257	80	131	801.6	421.1	2
258	58	128	158.8	4.3	2
259	71	140	606.3	175.1	2
260	70	121	30.7	0.0	2
261	79	138	730.1	1694.5	2
262	80	119	421.2	35.1	2
263	61	121	104.8	1.8	2
264	79	149	44.1	0	2
265	71	160	801.1	2535	2
266	78	159	742	3371.5	2
267	80	168	689.1	634.6	2
268	69	181	424.6	762.6	2
269	79	181	184.3	241.7	2
270	80	188	245.2	431.1	2
271	70	198	630	1992.1	2
272	81	200	0.0	0	1
273	100	48	48.7	0	2
274	80	49	757.4	473.8	2
275	90	58	739.8	280.2	2

276	88	39	520.7	76.8	2
277	100	60	0	0	2
278	80	59 0.0	0	0	2
279	101	38	730.5	464	2
280	101	68	383.1	97.8	2
281	79	70	508.8	103.9	2
282	90	79	573.3	138.3	2
283	100	78	372.4	70.9	2
284	81	81	585.8	197.2	2
285	100	91	397.2	38.9	2
286	80	89	614.5	192.3	2
287	91	99	734.9	159.6	2
288	101	100	599.3	539.3	2
289	81	98	181.2	1.3	2
290	98	111	744.8	1987	2
291	81	108	1022.3	643	2
292	90	120	899.3	1290.3	2
293	100	118	363.7	20.5	2
294	98	130	513.2	263.9	2
295	90	140	648.8	2147.5	2
296	90	138	645.4	1927.1	2
297	121	131	13	0.2	2
298	111	140	190.3	48.8	2
299	108	121	893	3070.9	2
300	120	141	104.7	14.6	2
301	119	118	150.4	16.9	2
302	158	228	558.4	551.6	2
303	140	229	558	513.9	2
304	150	241	318.5	129.2	2
305	151	218	394.3	239.2	2
306	161	241	141.9	8.6	2
307	141	240	112.5	4.6	2
308	160	218	580.4	1118.7	2
309	139	220	535.9	445.7	2
310	178	211	398.2	561.8	2
311	159	209	517.3	148.8	2
312	169	221	427.2	197.2	2
313	170	198	367.6	1429.8	2
314	180	218	374.7	77.5	2
315	178	201	144.8	81.3	2
316	158	198	169.8	154.5	2

317	219	88	235.1	136.9	2
318	198	90	611.7	735.8	2
319	211	100	746.4	710.6	2
320	208	80	436.6	495.8	2
321	221	99	540.9	586.8	2
322	199	98	801	1419	2
323	220	81	272.1	177.3	2
324	198	78	204.1	86	2
325	220	150	543.9	675	2
326	200	150	606.2	381.1	2
327	208	159	356	280.4	2
328	210	140	440.9	330.3	2
329	221	160	301.8	365.6	2
330	198	161	369.4	154.5	2
331	219	139	166.8	24.5	2
332	200	139	230.9	42.2	2
333	239	8	240.3	80.2	2
334	218	8	737.1	1373.4	2
335	229	19	518.6	147.7	2
336	239	18	390.7	186.7	2
337	218	18	797.4	1429.7	2
338	238	229	602.6	1510.9	2
339	218	228	430.8	265.2	2
340	231	239	354.1	478	2
341	230	221	602.4	538.9	2
342	240	239	172.6	51.9	2
343	221	241	324.8	290.9	2
344	239	220	420.1	398.3	2
345	218	218	763.5	1236.7	2
346	35	71	687.8	486.8	2
347	24	71	735.8	463.9	2
348	34	88	86.9	0.1	2
349	23	91	817	708.8	2
350	54	10	637.9	349.4	2
351	46	11	512.3	392	2
352	55	89	423.4	21.2	2
353	45	89	569.6	62.8	2
354	53	150	858	873	2
355	46	148	234	7.5	2
356	55	168	876	288.1	2
357	43	170	1082.8	1174.9	2

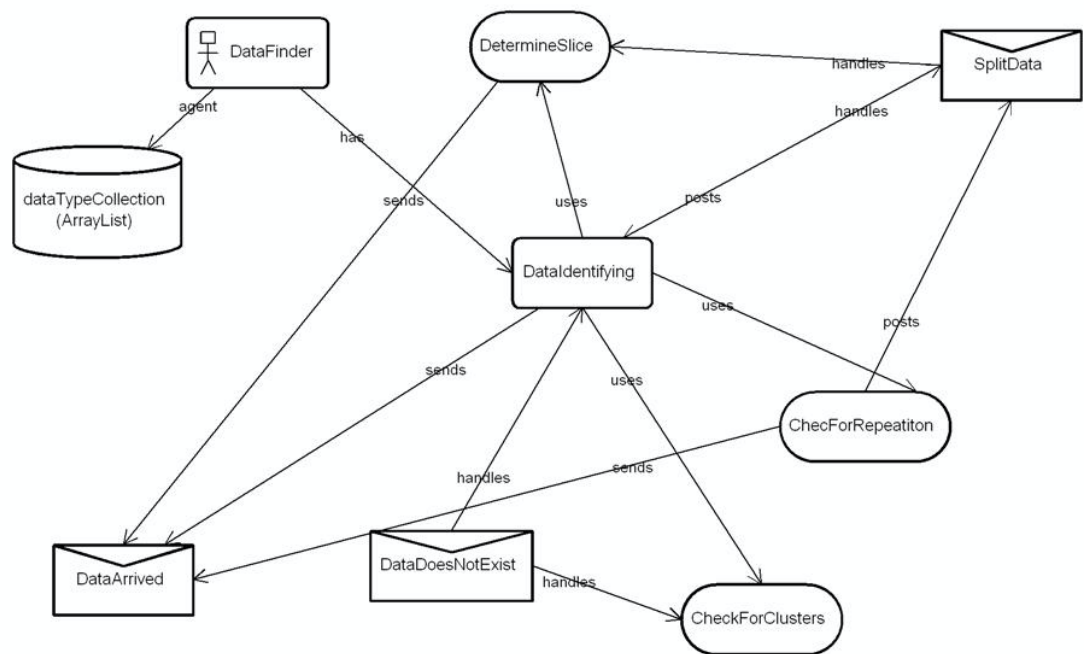
358	55	191	1392.6	1004.7	2
359	44	191	646.6	76.8	2
360	55	211	889.5	938.8	2
361	46	211	509.2	429.7	2
362	54	269	613.1	2922.4	2
363	43	271	767.8	198.1	2
364	73	29	649.4	231.7	2
365	64	31	235.4	10.8	2
366	75	129	782.8	639.3	2
367	64	129	227.3	8.6	2
368	73	149	722.9	696.1	2
369	64	151	974.5	664.1	2
370	75	171	512.2	144.8	2
371	63	168	1215.8	1446.1	2
372	73	188	687.1	2351.5	2
373	64	191	1259.9	1257.6	2
374	93	48	687.5	373.2	2
375	86	48	471.9	31.8	2
376	93	70	512.1	196.1	2
377	84	69	963.9	1210	2
378	93	90	874	1031.3	2
379	86	89	582.4	117	2
380	96	111	553.2	360.5	2
381	85	108	937.3	1495.5	2
382	93	131	883.6	1336.8	2
383	86	131	879.9	965.3	2
384	114	131	268.4	104.4	2
385	106	130	651.5	1957.7	2
386	155	229	386.4	88.2	2
387	145	230	333.2	63.5	2
388	174	208	339.2	335	2
389	166	211	600.3	647.6	2
390	215	89	595.2	1457	2
391	205	89	809.6	955.8	2
392	215	148	293.3	67.7	2
393	204	151	697.3	444.5	2
394	236	9	515.9	1593.8	2
395	223	9	613.2	277.6	2
396	236	229	665.3	1962	2
397	226	230	813.6	2279.8	2
398	35	80	174.8	2	2

399	24	79	891.8	635.7	2
400	36	61	699.6	1547.8	2
401	26	58	39.5	0.3	2
402	16	80	915.6	634.8	2
403	43	60	584	97.4	2
404	15	88	610	319.3	2
405	46	99	566.8	100.2	2
406	36	99	38.1	0	2
407	54	80	483	105	2
408	46	81	542.6	138.6	2
409	54	161	959.3	466.7	2
410	43	161	631.9	261.1	2
411	65	160	928.3	2252.5	2
412	33	160	431	48.6	2
413	36	170	672.3	605.5	2
414	53	179	1003.4	425.4	2
415	44	180	876.4	937.8	2
416	65	181	734.1	589.3	2
417	34	180	366	110.2	2
418	33	191	296.5	79.1	2
419	55	199	1069.2	376.4	2
420	46	198	804.3	674.6	2
421	63	201	731.1	1363.3	2
422	34	201	318.1	79.6	2
423	65	210	238.6	488.3	2
424	35	208	428.9	161.2	2
425	46	220	734.4	1236.1	2
426	36	219	429.1	152.3	2
427	35	217	597.4	397	2
428	53	258	442.6	1696.4	2
429	46	260	765.2	779.8	2
430	45	281	605.5	934.8	2
431	35	278	795.9	1588.3	2
432	35	259	235	18.9	2
433	84	30	562	85.3	2
434	84	41	411.4	34	2
435	75	40	696.7	356.2	2
436	73	141	790.9	607.3	2
437	63	140	696.5	357.8	2
438	84	138	687.3	893.4	2
439	76	159	597.5	997.3	2

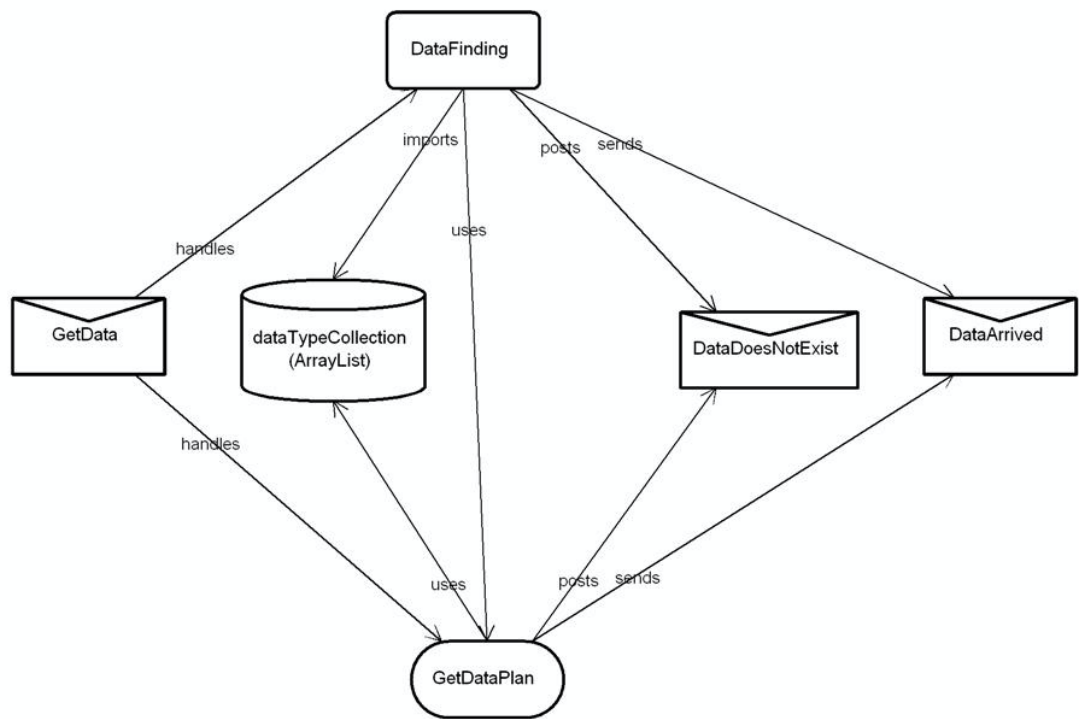
440	84	161	437.4	387.2	2
441	86	169	317.4	761.7	2
442	73	199	470.7	5190.1	2
443	76	51	498.7	101.8	2
444	94	61	778.7	1354	2
445	85	60	523.3	117.5	2
446	104	38	617.1	200.2	2
447	93	41	395.5	28.4	2
448	75	90	518.9	113	2
449	94	101	383.7	12.9	2
450	85	100	704.1	126	2
451	104	109	562.3	908.6	2
452	75	110	655.3	349	2
453	95	121	823.6	548.4	2
454	83	119	847.7	701.4	2
455	94	140	607.5	723.2	2
456	103	139	491.2	565.3	2
457	114	120	319.5	154.2	2
458	104	118	594	289.2	2
459	196	91	433.5	254.1	2
460	215	101	209.6	4	2
461	204	101	533.8	127.3	2
462	196	101	592.4	419.4	2
463	195	149	478.7	141.9	2
464	216	11	660.2	1424.8	2
465	225	19	832.2	512.2	2
466	214	19	242.5	15.6	2
467	245	231	161.2	26.1	2
468	233	220	626	959.7	2
469	226	221	800.1	1681.5	2
470	213	218	482.6	476.2	2

APPENDIX C: AGENT DESIGN FROM JACK CONSOLE

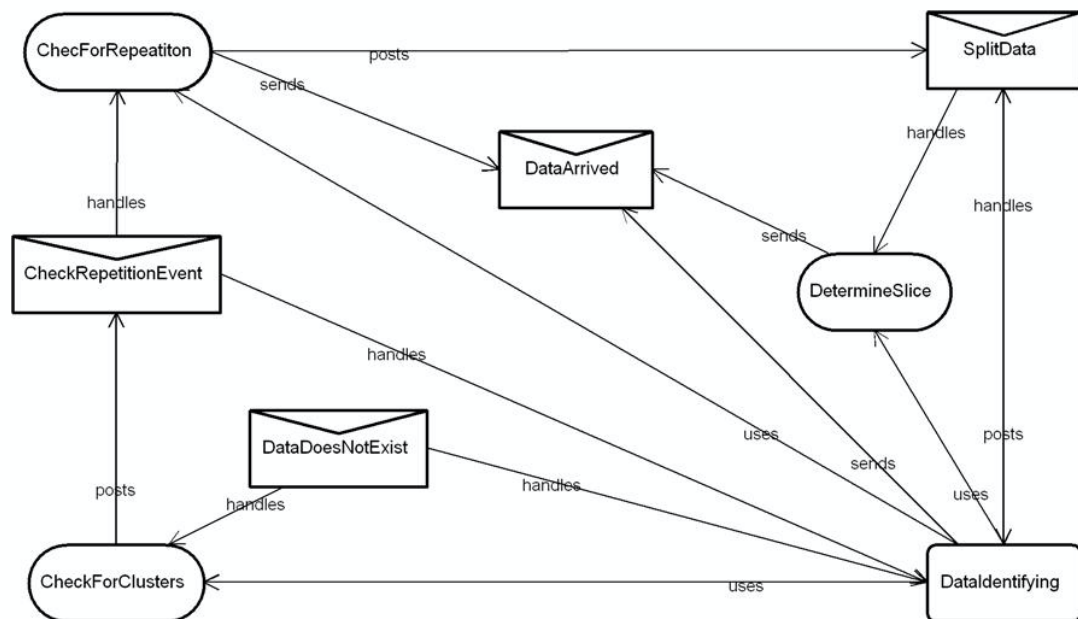
Design: DataFinder_AD



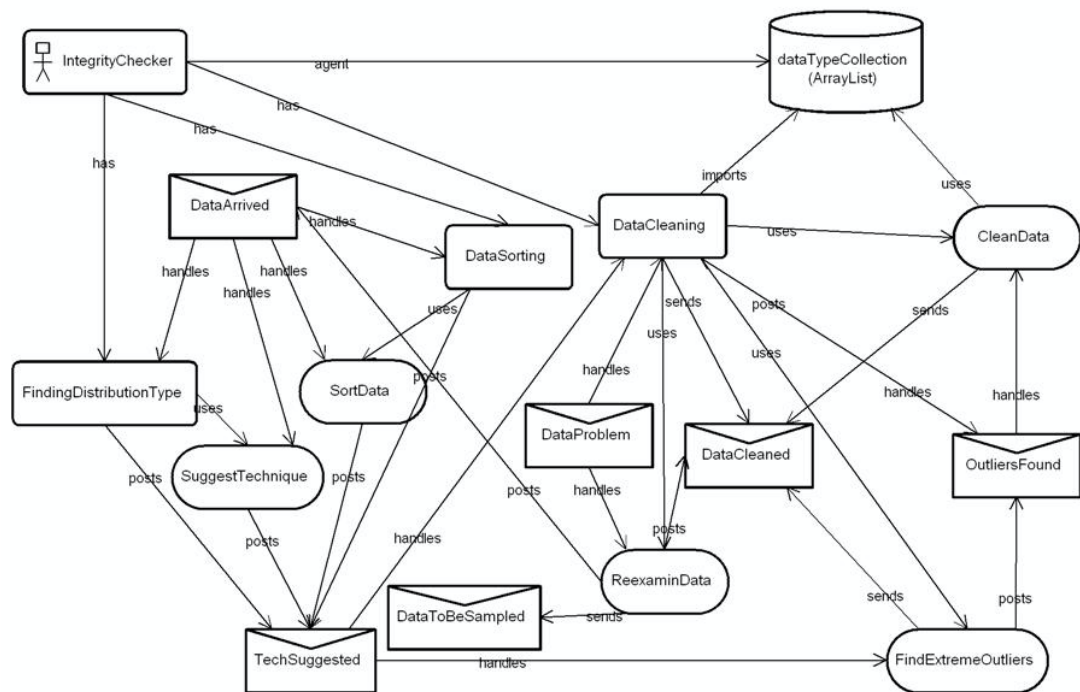
Design: DataFinding_CAP



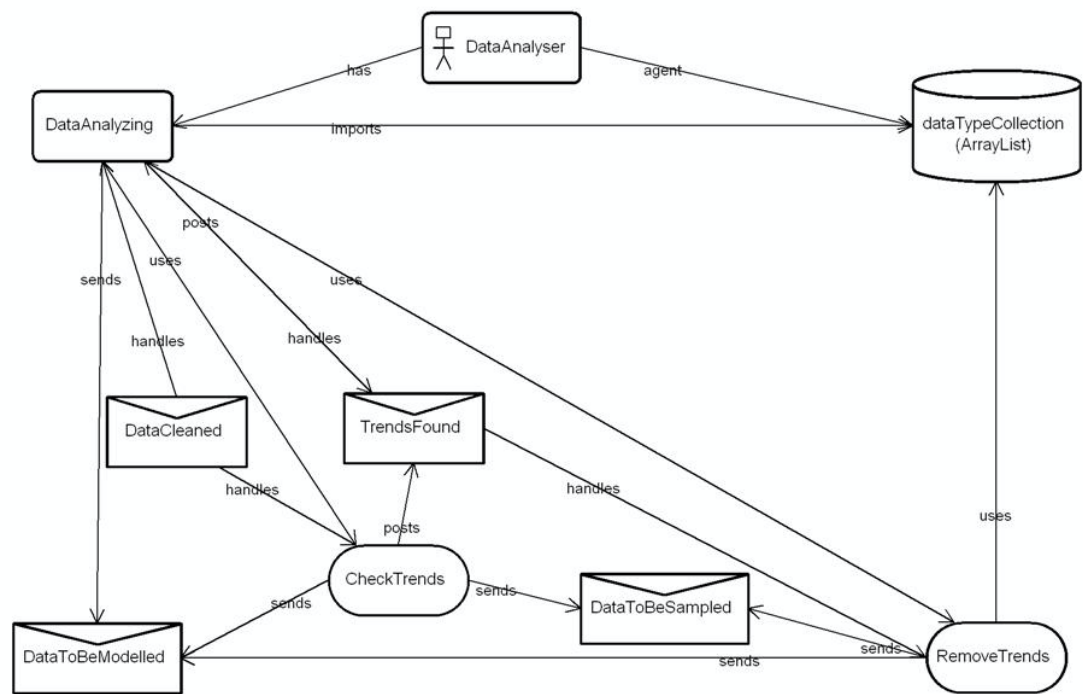
Design: DataIdentifying_CAP



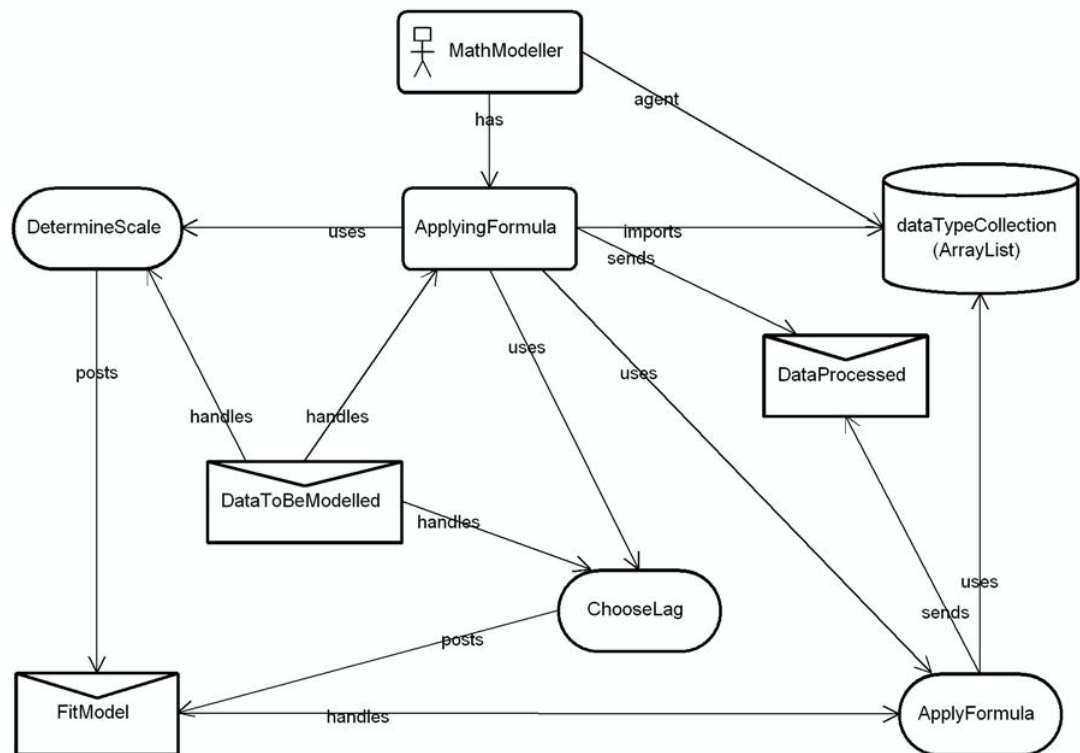
Design: IntegrityChecker_AD



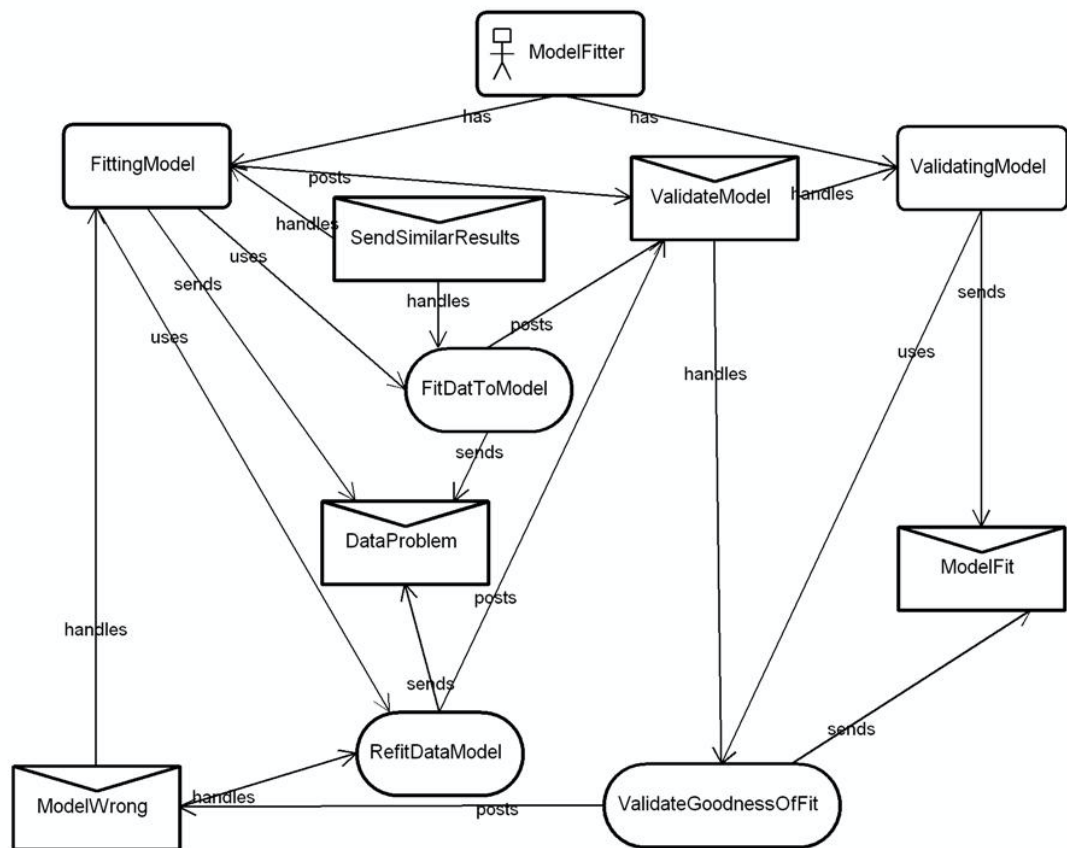
Design: DataAnalyser_AD



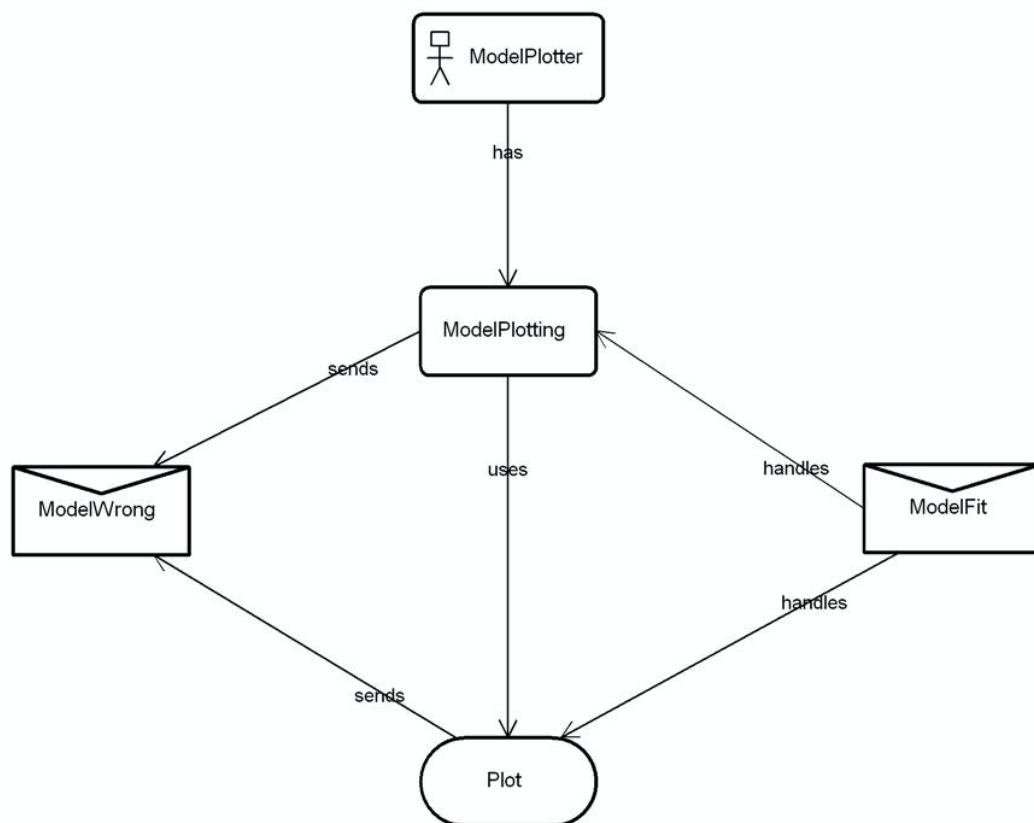
Design: MathematicModeller



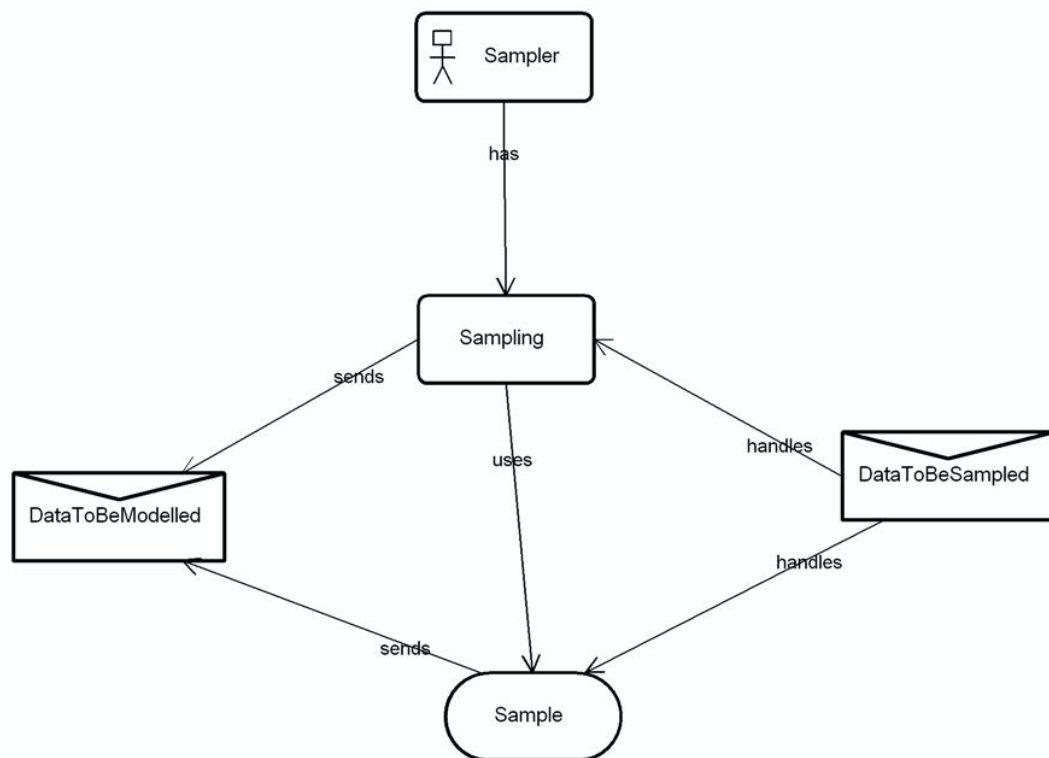
Design: ModelFitter_AD



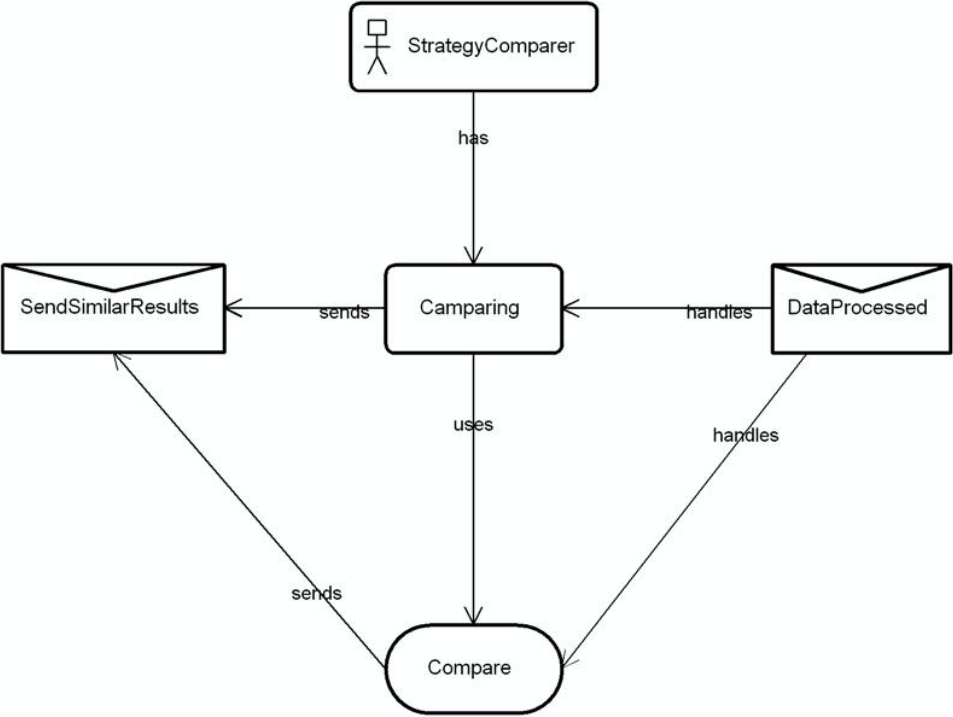
Design: ModelPlotter_AD



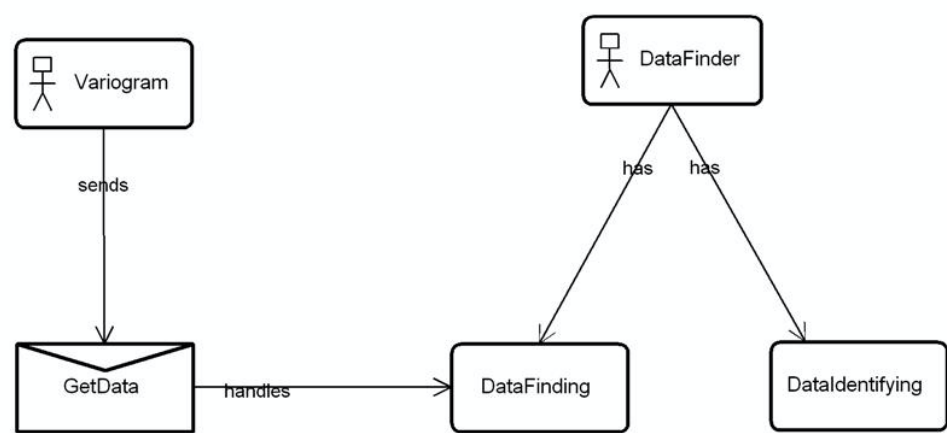
Design: Sampler



Design: StrategyComparer



Design: Variogram_AD



APPENDIX D: VARIAOGRAM AND CROSS VALIDATION DATA

Anisotropic at 0°

1	10.07	51120.7	68
2	19.73	62261.04	215
3	30.35	75819.95	174
4	42.2	65093.42	386
5	56.73	79103.28	492
6	68.73	106599.5	433
7	80.05	83655.71	436
8	91.44	93359.62	403

Anisotropic at 45°

1	10.37	57283.11	42
2	16.39	76487.4	131
3	28.91	78870.65	284
4	42.51	95868.4	234
5	56.16	97050.96	302
6	68.84	93038.22	235
7	81.19	91570.6	302
8	92.64	91445.42	272

Anisotropic at 90°

1	7.62	54424.08	133
2	18.61	69377.91	330
3	30.52	103157.5	295
4	42.41	91154.07	324
5	56.46	86843.68	267
6	67.64	90220.71	181
7	80.33	64516.8	211
8	93.16	94716.91	168

Anisotropic at 135°

1	9.89	48215.13	32
2	15.87	65240.34	146
3	28.92	80526.39	310
4	42.93	81180.72	285
5	56.53	76867.69	369
6	68.72	84050.52	300
7	80.9	102112.7	341

8	92.45	109066	276
---	-------	--------	-----

Experimental variogram

1	7.62	54424.08	133
2	18.61	69377.91	330
3	30.52	103157.5	295
4	42.41	91154.07	324
5	56.46	86843.68	267
6	67.64	90220.71	181
7	80.33	64516.8	211
8	93.16	94716.91	168

Variogram with Gaussian fit

1	8.03	46301.12	739
2	16.92	67792.5	2551
3	27.89	84853.73	3249
4	39.11	88045.77	3687
5	49.65	89372.38	4134
6	60.69	88568.11	4991
7	71.5	90414.52	5065
8	82.32	87823.68	5485

Variogram with spherical fit

1	7.1	40478.56	523
2	14.81	68748.92	2035
3	24.66	77775.37	2839
4	34.63	91312.23	3057
5	44.6	84768.83	3804
6	54.79	90462.5	4032
7	64.5	89683.38	4632
8	74.58	90926.1	4877

Cross validation

1	0	252.96
4	434.4	564.88
5	412.1	581.6
6	587.2	465.42
8	31.3	399.39
9	388.5	318.42
12	82.1	224.22
13	81.1	226.12
14	124.3	276.31
16	28.7	117.28
17	78.1	135.31
18	292.1	292.48
19	895.2	677.39
20	702.6	496.93
21	490.3	458.84
22	136.1	352.26
23	335	297.06
24	277	491.58
25	206.1	309.42
26	24.5	419.31
27	198.1	284.18
28	60.3	250.21
29	312.6	530.39
30	240.9	487.44
31	653.3	512.52
32	96.4	144.41
33	105	242.74
34	37.8	372.51
35	820.8	527.7
36	450.7	387.96
37	190.4	293.1
38	773.3	591.39
39	971.9	912.63
40	762.4	919.32
41	968.3	696.95
42	394.7	394.54
43	343	405.16
44	863.8	676.53
45	159.6	343.55
46	445.8	360.6

47	673.3	515.74
48	252.6	390.56
49	537.5	332.98
50	0	473.83
51	329.1	371.49
52	646.3	337.03
53	616.2	810.65
54	761.3	820.81
55	918	809.21
56	97.4	317.38
60	2.4	196.87
61	368.3	400.54
62	91.6	539.15
63	654.7	594.02
64	645.5	646
65	907.2	772.52
66	826.3	723.04
67	975.3	876.95
68	551.1	423.13
69	155.5	277.57
70	10.7	175.47
73	0	152.01
74	12.1	145.84
75	62.2	134.81
76	399.6	354.57
77	176.6	412.75
78	402	182.08
79	260.6	312.23
80	192	371.89
81	237.6	446.06
82	702	548.16
83	38.5	258.08
84	22.1	124.45
85	2.7	80.87
86	17.9	117.49
87	174.2	149.15
88	12.9	155.25
89	187.8	108.65
90	268.8	109.42
91	572.5	260.03

92	29.1	289.27
93	75.2	304.06
94	399.9	307.81
95	243.1	331.58
96	0	286.82
97	244.7	77.24
98	185.2	92.69
99	26	107.16
100	0	122.45
101	100.3	234.08
102	530.3	294.67
103	107.4	203.32
104	159.3	141.6
105	70.7	143.27
106	260.2	387.67
107	326	264.33
108	332.7	288.48
109	531.3	330.5
110	547.2	382.59
111	482.7	281.13
112	84.1	256.28
113	4.7	159.12
114	180.6	105.97
116	342.4	340.23
117	602.3	376.95
118	209.1	181.96
119	79.4	163.03
120	104.1	95.32
121	446	302.98
122	189.9	287.37
123	280.4	262.1
125	499.3	418.01
126	457.3	376.26
127	341.2	217.3
128	0	258.16
129	208.3	172.96
130	99.7	254.59
131	636.6	467.58
132	173.1	334.52
133	17	178.52
134	283.1	80.47
135	30.9	121.34
136	348.5	324.52

137	222.4	290.16
138	59.1	264.86
139	0	275.04
140	326	393.5
141	325.1	397.02
142	114.7	308.58
143	481.6	332.86
144	324.1	222.42
145	10.9	201.24
146	332.9	254.43
147	184.4	295.13
148	146.6	191.32
149	92	120.96
150	2.5	90.28
151	358.1	483.68
152	473.3	303.96
153	308.8	297.92
154	406.8	270.24
155	812.1	706.74
156	339.7	379.61
157	223.9	297.18
158	673.5	522.84
159	141	280.37
160	61.8	233.94
161	258.3	398.33
162	590.3	380.27
163	166.9	232.76
164	125.2	117.54
165	29.3	78.08
166	617.6	545.92
167	425.9	450.26
168	295.7	359.35
169	224.9	292.95
170	31.7	284.56
171	377.4	240.69
172	333.3	205.52
173	351	319.21
175	137.6	356.7
176	451.2	456.96
177	639.5	737.02
178	119.9	222.53
179	27.2	111.64
180	2.1	101.4

181	167.7	304.96
182	147.8	383.81
183	442.7	311.8
184	487.7	255.26
185	0	256.8
186	28.2	183.22
187	0	190.31
188	18.3	244.29
189	266.3	251.72
190	502.3	275.35
192	240.9	238.83
193	234.4	168.59
194	22.4	139.21
195	45.6	105.34
196	76.2	462.1
197	284.3	684.45
198	606.8	660.08
199	772.7	297.81
200	269.5	331.14
201	1036.7	871.41
202	238.6	612.99
203	519.4	250.41
204	414.9	373.82
205	601.4	779.79
206	579.2	321.42
207	601.4	330.56
208	594.6	599.52
209	550.1	464.33
210	99.4	285.61
211	233.6	305.26
212	14.4	273.11
213	115.9	150.04
214	506.2	330.77
215	502.4	499.58
216	608	533.7
217	363.9	410.29
218	385.6	421.75
219	1521.1	897.92
220	340.9	330.85
221	879.1	859.22
222	413.4	348.48
223	868.9	1065.33
224	657.4	556.11

225	477	629.56
226	268.5	280.78
227	806.4	1087.4
228	914.4	825.77
229	811.5	958.72
230	1113.6	939
231	1008	667.3
232	1528.1	1286.7
233	970.9	589.59
234	1109	964.02
235	1203.9	1034.57
236	641.3	593.11
237	720.6	737.52
238	665.3	492.05
239	543.3	604.39
240	101.1	499.78
241	615.9	500.07
242	543.1	422.43
243	868.8	730.44
244	583	555.61
245	670.7	581.43
246	148.8	333.1
247	798	709.11
248	194.9	451.27
249	635.2	405.44
250	781.6	580.58
251	238.6	141.31
252	472	457.99
253	58.1	446.49
254	600.3	619.49
255	64.9	263.42
256	505.9	409.58
257	801.6	789.1
258	158.8	220.09
259	606.3	753.05
260	30.7	454.19
261	730.1	689.05
262	421.2	729.21
263	104.8	206.84
264	44.1	673.54
265	801.1	751.74
266	742	542.14
267	689.1	466.62

268	424.6	657.46
269	184.3	348.35
270	245.2	275.34
271	630	639.06
272	0	334.49
273	48.7	514.72
274	757.4	471.57
275	739.8	630.98
276	520.7	403.65
277	0	507.76
278	0	540.11
279	730.5	484.69
280	383.1	310.59
281	508.8	681.18
282	573.3	597.69
283	372.4	429.15
284	585.8	578.56
285	397.2	551.83
286	614.5	505.61
287	734.9	568.19
288	599.3	424.37
289	181.2	664.82
290	744.8	531.19
291	1022.3	757.48
292	899.3	843.7
293	363.7	686.59
294	513.2	754.73
295	648.8	617.4
296	645.4	696.38
297	13	212.27
298	190.3	250.7
299	893	510.77
300	104.7	135.06

301	150.4	220.7
302	558.4	377.47
303	558	389.33
304	318.5	212.88
305	394.3	514.19
306	141.9	232.5
307	112.5	319.23
308	580.4	493.53
309	535.9	408.07
310	398.2	350.66
311	517.3	444.58
312	427.2	434.95
313	367.6	205.92
314	374.7	349.98
315	144.8	286.27
316	169.8	331.09
317	235.1	418.78
318	611.7	511.83
319	746.4	397.18
320	436.6	467.04
321	540.9	240.18
322	801	584.61
323	272.1	290.17
324	204.1	340.13
325	543.9	301.15
326	606.2	556.66
327	356	421.06
328	440.9	294.98
329	301.8	349.08
330	369.4	384.24

APPENDIX E: COMPARING AGENT DEVELOPMENT SYSTEMS

Platform	Primary Domain	Programming Engine	Operating System	FIPA Compliant	GIS Capabilities
A3 / AAA (Agent Anytime Anywhere)	General-purpose distributed and atomic agent based platform.	Java	Any Java Virtual Machine.	N/A	N/A
ABLE (Agent Building and Learning Environment)	Building intelligent agents using machine learning and reasoning	Able Rule Language (ARL)	OS/2; Windows 95; Windows 98; Windows NT; Java 2 JVM)	N/A	N/A
AgentBuilder	General purpose multi-agent systems	Knowledge Query and Manipulation Language (KQML); Java; C; C++	Any platform with a Java Virtual Machine	N/A	N/A
AnyLogic	Agent-based general purpose; distributed simulations	Java; UML-RT (UML for real time)	Applets and Java Virtual Machine	N/A	Yes
Ascape	General-purpose agent-based models.	Java	Windows; Macintosh; Unix; Linux; web	N/A	N/A
Brahms	Multi-agent environment for simulating people's activity and situated behavior (location, artifacts, communication, etc.). Used for modeling and simulating work practice in organizational processes.	The BVM (Brahms Virtual Machine) is a multi-agent discrete-event engine, running each agent as a separate event-based Java thread. Brahms and Java agents can interact together easily.	Windows 2000; Windows XP; Linux; Sparc/Intel Solaris; and Mac OS X	Yes, agents use Communicative Acts objects to send FIPA messages.	Yes: Hierarchical user defined objects that can have attributes represent coordinate, etc.
Breve	Building 3D simulations of multi-agent systems and artificial life.	Simple Interpreted object oriented language called Steve; agent behaviors can be written in python	Mac OS X; Linux; and Windows	N/A	N/A
JADE	Distributed applications	Java	Any Java Platform	Yes	N/A

	composed of autonomous entities				
JCA-Sim	Cellular automata; General purpose simulator	Java; Cellular Description Language (CDL) (for input to simulation) Java/Jess/JessML (declarative xml rule language)	Any Java Platform	N/A	N/A
JESS	Rule engine and scripting environment Social science; domain specific programming language for developing agent based models		Java Virtual Machine	N/A	N/A
MAML (Multi-Agent Modeling Language)	General purpose; social complexity, physical modeling, abstract modeling, AI/machine learning	MAML language; C; visual programming interface	PC; Linux	N/A	N/A
MASON		Java FABLES (Functional Agent-based Language for Simulations); Java; it is possible to run Repast and NetLogo simulations too.	Any Java Platform (1.3 or higher)	N/A	Yes
MASS (Multi-Agent Simulation Suit)	General purpose, distributed simulations, participatory simulations.		Any OS with Java 1.5, tested for Windows, MacOSX, Linux Java version 1.4, although a 1.3 version for Mac OS X is available. To run the demonstration simulations, you'll need a Java Runtime Environment (RepastS, RepastJ); platform independent (RepastPy); Windows (Repast.net)	No	No
Repast	Social sciences	Java (RepastS, RepastJ); Python (RepastPy); Visual Basic, .Net, C++, J#, C# (Repast.net)		N/A	Yes
SDML (Strictly	Multi-agent systems (with	Smalltalk release 5i.2	Windows 3.1; Windows 95;	N/A	N/A

Declarative Modeling Language)	limited rationality)	Non-Commercial	Windows 98; Windows 2000; Windows NT; Linux; Intel; PowerMac; Unix; ADUX/AIX/HPUX/ SGI/Solaris		
SEAS (System Effectiveness Analysis Simulation)	The US Air Force's Multi-Agent Theater Operations Simulation	Tactical Programming Language (TPL)	32-bit and 64-bit Windows 2000/XP/Vista/7	N/A	N/A
SeSAM (Shell for Simulated Agent Systems) (fully integrated graphical simulation environment)	General purpose multi domain (agent based); research, teaching, resources, graph theory	Simulation compiled from visual specification; Visual programming	Java 5.0 or better; Windows; Linux; Mac OS X Available for Meiko and BBN multi-computer systems and can be used on a network with Sun3, Sun 4, and HP 9000 workstations	Plugin available	Raster- and Vector-GIS as spatial representation, ESRI-Arcview
Jade's sim++	Parallel simulation; Applied simulations; network planning; electronic CAD; real time communication simulation	C++	Any Java Platform	N/A	N/A
JIAC	General purpose	Java		Yes	N/A
Spatial Modeling Environment (SME)	Ecological economic; Ecosystems modeling	No knowledge of computer programming required	Unix	N/A	N/A
Swarm	General purpose agent based Multi-agent simulator in	Java; Objective-C	Windows; Linux; Mac OS X	N/A	N/A
VisualBots	Microsoft Excel	Visual Basic	Windows	N/A	Yes
ZEUS	Rules engine and scripting environment; Distributed multi-agent simulations	Visual editors and code generators	Windows 95; Windows 98; Windows NT; Windows 2000; Windows XP; Linux; BSD; UNIX-like OSes; Solaris	Yes	N/A

(Source: amended from http://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software)

APPENDIX F: THE AGENT KNOWLEDGE ACQUISITION, THE LOG FILE FOR THE WALKER DATA

DF: New File Found in dulaFolder1 in Server IP 127.0.0.1
DF: Data unknown process and strategy to be analysed
DF: Analysis log is being used to speed up variogram development process
DF: Data size too large
DF: data Split
DF: Data subsets are stored for separate use
DF: 3 clusters found
DF: Data archived and stored to memory
Agent Data Analyser initiated
DA: New data received from DataFinder
DA: Checking for trends
DA: Timeout-error occur during trend analysis due to algorithms unavailable
DA: No trend in data
DA: Data archived and stored to memory
Agent IntegrityChecker initiated
IC: Data has been sorted in ascending order for easy analysis
IC: Suggested technique for this data is robust
IC: Data that were determined erroneous has been removed and saved in file err.dat for expert opinion
IC: 3 Outlier found
IC: 3 Outlier found
IC: Saved on file name outlier1.dat
IC: Saved on file name outlier2.dat
IC: Saved on file name outlier3.dat
IC: Failed and aborted
-IC: Data clean and integrity confirmed
IC: Data archived and stored to memory
Agent Sampler initiated
S: Dataset large and has been sampled affectively
S: No enough data for efficient modeling
S: Sampling strategy used with 124 datum being held (inclusive)
S: Sampling strategy used with 250 datum being held (inclusive)
S: Sampling strategy used with 353 datum being held (inclusive)
S: 353 data ready for modeling
S: Data archived and stored to memory
Agent MathematicalModeller initiated
MM: Experimental Variogram created for Data err1.dat
MM: Experimental Variogram created for Data Sample1.dat
MM: Experimental Variogram created for Data Sample2.dat
MM: Experimental Variogram created for Data Sample3.dat
MM: Experimental Variogram created for Data Walker.dat at 0 degree
MM: Experimental Variogram created for Data Walker.dat at 45 degree
MM: Experimental Variogram created for Data Walker.dat at 90 degree

MM: Experimental Variogram created for Data Walker.dat at 135
 degree
 MM Scale of 2 is determined
 MM: Lag of 10 is determined
 MM: Data archived and stored to memory
 Agent StrategyComperer initiated
 SC: Model of created variograms (sample) results have been
 compared SC: Model values at 4/10
 -SC: Failed and aborted
 SC: Initiate Dynamic binding
 SC: User reinitiated Agent DataAnalysis
 SC: Model values at 9/10
 -SC: Pass
 SC: Data archived and stored to memory
 Agent ModelFitter initiated
 MF: Examining defined lag and scale
 MF: Gaussian model applied
 -MF: Spherical model applied
 MF: Kriged determine good fit
 -MF: Kriged data too large require extra resources for
 processing, data saved on krig1.dat
 MF: Cross validation pass, data saved on cross1.dat
 -MF: Cross validation fail, data saved on cross1.dat
 MF: Goodness of fit determined, pass
 MF: Goodness of fit determined, fail
 MF: bad fit, refit
 MF: Use data for graphics
 MF: Data archived and stored to memory

 From each agent termination @@Process has taken 09:23

APPENDIX G: PUBLISHED MATERIALS

Al-Zakwani, A. (2006) Mobile Devices Evolution and Revolution: A Cause for Security Concern. *International Conference on Internet Computing* **2006**: 369-376

Al-Zakwani, A. (2007a) Establishing a platform for networked games on mobile devices using SMS/GPRS, *Proceedings of Advances in Computing and Technology, (AC&T) The School of Computing and Technology 2nd Annual Conference*, University of East London, pp.128-134

Al-Zakwani, A., Brimicombe, A. and Mouratidis, H. (2007b) An Agent-Based System to Support Geo-Information Analysis. *IAT* **2007**: 269-272

Al-Zakwani, A., Mouratidis, H. and Brimicombe, A. (2007a) "An agent-based system to support geo-information analysis" *Proceedings Intelligent Agent Technology (IAT 2007)*, Silicon Valley, CA: 269-272

Al-Zakwani, A., Mouratidis, H. and Brimicombe, A. (2007b) "A dynamic binding technology for agent-based geo-information systems" *Proceedings of Intelligent Systems & Agents (IADIS 2007)*, Lisbon, Portugal: 117-123 [note: Outstanding Paper Award]